NumPy -da Təsadüfi Ədədlər, Random Data Distribution və Seaborn – Part 5

Proqramçılar Random ədədi adətən "təsadüfi ədəd" və ya "həmişə fərqli bir ədəd" kimi tanıyır, lakin məşhur inancın əksinə olaraq, təsadüfi sözü bunu ifadə etmir. Random "məntiqi olaraq gözlənilməz" deməkdir.

Digər tərəfdən, təsadüfilik üçün "məntiqi olaraq gözlənilməz" məsələsi bir qədər qarışıqdır. Çünki, kompüterlərlə işləyən bütün proqram kodları müəyyən bir alqoritm əsasında işləyir. Təsadüfi ədədlər yaradılarkən, müəyyən bir alqoritm vasitəsilə çalışarlar. Bu halda təsadüfi ədədi yaradan proqram kodu proqnozlaşdırıla biləndir, ona görə də təsadüfün gözlənilməz olduğunu söyləmək məntiqi cəhətdən düzgün görünmür.

Bu səbəbdən təsadüfi ədədlər yaratmaq üçün istifadə edilən alqoritmə psevdo-təsadüfi (yalançı təsadüfilik) deyilir. Beləliklə, həqiqətən gözlənilməz rəqəmlər yarada bilərikmi?

Bəli. Əgər kompüterlərimizdə proqnozlaşdırıla bilməyən real təsadüfi ədədlər yaratmaq istəyiriksə, bizə kənar mənbələrdən təsadüfi məlumat lazım olacaq. Xarici mənbələrdən alınan məlumatlara klaviatura düymələri, siçan hərəkətləri, şəbəkədən alınan məlumatlar və s. ola bilər. Həqiqi təsadüfi ədədlər təhlükəsizlik açarları və parollar kimi həssas nöqtələrdə və rəqəmsal rulet masaları kimi tətbiqlərdə lazımdır.

Bu bölmədə təsadüfi ədədlərdən danışarkən real təsadüfi ədədlərdən deyil, psevdo-təsadüfi ədədlərdən istifadə edəcəyik.

Təsadüfi bir ədədi necə yaratmaq olar? NumPy-də təsadüfi ədədlər yaratmaq üçün random modulu var. Bu modul vasitəsilə müəyyən intervallarla tam (int), onluq (float) və ya massiv (array) şəklində təsadüfi ədədlər yaratmaq mümkündür.

rand() funksiyası 0-1 arasında təsadüfi ədəd yaradar (1 daxil deyil).

randint(100) funksiyası 0-100 (100 daxil deyil) arasında təsadüfi tam ədəd qaytarır.

randint(x,y) funksiyası müəyyən x və y aralığı diapazonunda təsadüfi tam ədədlər yaradar. x və y də mənfi qiymətlər qəbul edə bilər.

```
from numpy import random
import numpy as np

# 0 ilə 1 arasında təsadüfi ədəd yarad (onluq)
sayi1 = random.rand()
print(sayi1)

# 0 ilə 100 arasında təsadüfi ədəd yarat (tam ədəd)
sayi2 = random.randint(100)
print(sayi2)

# 50 ilə 150 arasında təsadüfi ədəd yarat (tam ədəd)
sayi3 = random.randint(50,150)
print(sayi3)
```

```
0.7026050655453164
79
63
```

NumPy-də massivlərlə işlədiyimiz üçün təsadüfi massivlər yaratmaq üçün randint() funksiyası ilə massivin formasını (shape) təyin edə bilərik.

```
print(random.randint(10, size=4))
# 0 ilə 4 arasında tam ədədlərdən ibarət
# 2D, 4 elementli (2x4 ölçüsündə) bir massiv yaradaq
print(random.randint(5, size=(2, 4)))
```

```
[7 8 0 1]
[[0 3 0 0]
 [2 2 3 3]]
```

Biz rand() funksiyası ilə də daxilində sıfır və bir aralığında ədədlər tutan massivin formasını təyin edə bilərik.

```
# 1D massiv içində 5 ədəd onluq ədəd olsun
print(random.rand(5))

# 2D və 3x5  ölçüsündə daxili onluq ədədlərdən ibarət massiv yaradaq
print(random.rand(3 , 5))

# Diqqət! random() və randint() fərqlidir
print(random.randint(3 , 5))
```

```
[0.43959587 0.3007422  0.90586985 0.80258224 0.27578968]
[[0.32477243 0.90539095 0.75025628 0.55162352 0.75492504]
 [0.49940115 0.60569508 0.20680367 0.74776982 0.83246233]
 [0.12394195 0.08762053 0.46821164 0.89808327 0.71653689]]
3
```

Başlanğıc (x) və son (y) dəyərləri olaraq, randint(x,y) funksiyasına birdən çox qiymət - dəyər təyin etmək mümkündür.

```
# 1D, 3 elementli, 3 fərqli üst sərhədi olan massiv yarad
print(random.randint(1,[5,7,10]))
# 1D, 3 elementli, 3 fərqli alt sərhədi olan massiv yarad
print(random.randint([3,5,10] , 20))
# 2D, 4 elementli, broadcasting istifadə edərək
# bir  olan massiv yaradaq
print(random.randint([3, 5, 7, 9],[[20],[40]]))
```

```
[2 3 6]
[17  7 11]
[[ 4 11 19 17]
 [27 26 30 11]]
```

- Massivdən təsadüfi ədədin əldə edilməsi

Choice() metodu sizə massivdən təsadüfi dəyər əldə etməyə imkan verir. Choice() massivi parametr kimi götürür və təsadüfi olaraq dəyərlərdən birini qaytarır. Choice() ilə biz dəyərlər massivi də qaytara bilərik. Bu halda massivin shape -sini təyin etmək üçün size parametrinə tuple şəklində dəyərlər əlavə edilər.

```
#choice ilə təsadüfi ədəd
# verilən massivin dəyərlərindən birini qaytarar
print(random.choice([3, 26, 35, 57]))
# verilən dəyərlərlə 3x5 ölçüsündə 2D massiv yaradar
print(random.choice([3,26,35,57], size = (3,5)))
```

```
3
[[ 3  3 26  3 26]
 [57  3 57  3 35]
 [57 57 26  3 57]]
```

- Random data disturbution - verilənlərin təsadüfi paylanması

Veri dağılımı, tüm olası değerlerin ve her bir değerin ne sıklıkta oluştuğunun bir listesidir. İstatistik ve veri bilimiyle çalışırken bu tür listeler sıklıkla kullanılır. NumPy Random modülü, random olarak oluşturulmuş veri dağılımlarını döndüren yöntemler sunmaktadır.

Verilənlərin paylanması bütün mümkün dəyərlərin və hər bir dəyərin nə qədər tezlikdə (frequency) baş verdiyini ifadə edən siyahıdır. Bu cür siyahılar tez-tez statistika və data elmi ilə işləyərkən istifadə olunur. NumPy Random modulu təsadüfi formada yaradılan data paylamalarını ifadə edən üsulları təmin edir.

Random paylama müəyyən bir ehtimal sıxlığı funksiyasına (probability density function) uyğun təsadüfi ədədlər çoxluğudur.

Ehtimal sıxlığı funksiyası: Kəsilməz ehtimalı təsvir edən funksiyadır. Məsələn, massivdəki bütün dəyərləri əldə etmək ehtimalı.

Random modulun choice() funksiyası müəyyən edilmiş ehtimallara əsaslanaraq təsadüfi ədədlər yaradır. Həmçinin burada hər bir dəyər üçün ehtimalı müəyyən edə bilərik. Ehtimalı 0 və 1 arasında olan rəqəmlə müəyyən edilir: 0 (sıfır) dəyərin heç vaxt baş verməyəcəyini, 1 isə dəyərin həmişə baş verəcəyini göstərir.

```
# random data disturbution - verilənlərin təsadüfi paylanması

# 1, 3, 5, 7 dəyərlərindən ibarət 3x5 ölçüsündə 2D massiv yaradaq
# Dəyərin 1 olma ehtimalı 0.1
# Dəyərin 3 olma ehtimalı 0.3
# Dəyərin 5 olma ehtimalı 0.6
# Dəyərin 7 olma ehtimalı 0 olsun

print(random.choice([1, 3, 5, 7], p=[0.1, 0.3, 0.6, 0.0], size=(3, 5)))
```

```
[[5 5 1 5 3]
 [5 3 3 3 3]
 [5 5 5 5 3]]
```

Bütün ehtimal ədədlərinin cəmi 1 olmalıdır. Yuxarıdakı nümunədə göründüyü kimi, 0,1+0,3+0,6+0,0=1,0

Digər tərəfdən, nə qədər qaytarılmasından asılı olmayaraq, bu massivdə heç vaxt 7 dəyəri olmayacaq. **"Size"** parametri massivin shape -sini təyin edir, biz burada sabit dəyər, məsələn, 10 göstərməklə 10 ədəd elementi olan 1-D massivi də əldə edə bilərik.

- Random - Təsadüfi Permutasiyalar

Permutasiya massivdəki elementlərin düzülüşünə aiddir. Misal üçün. [3, 2, 1], [3, 1, 2], [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1] düzülüşü permutasiyanı ifadə edər.

NumPy -ın Random modulu bu məqsədlə bizi iki funksiya ilə təmin edir: shuffle() və permutation().Shuffle() massivin özündə ardıcıllığın dəyişdirilməsinə xidmət edir. Beləliklə, shuffle() orijinal massivdə dəyişikliklər edir.

```
# randomly shuffle
massiv = np.array([1,3,5,7])
random.shuffle(massiv)
print(massiv)
```

```
[5 3 7 1]
```

```
# random permutasiya
massiv = np.array([1,3,5,7])
print(random.permutation(massiv))
```

```
[1 3 5 7]
```

Permutation() isə dəyişikliklər edilmiş yeni bir massiv qaytarır və beləliklə orijinal massiv olduğu kimi qalır.

- Seaborn: statistik verilənlərin vizuallaşdırılması

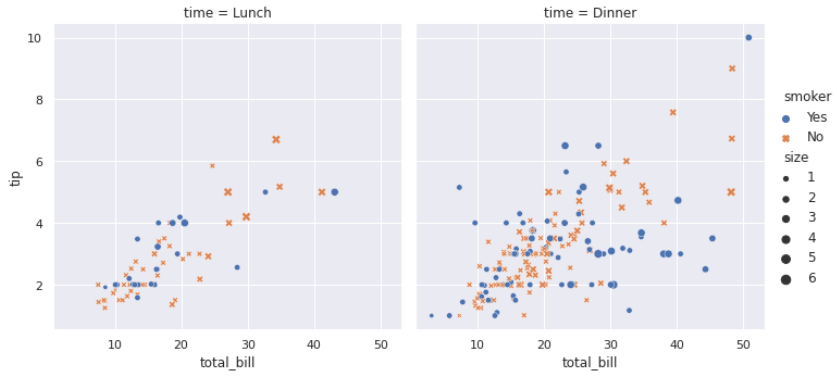
Seaborn matplotlib təməlli Python data vizuallaşdırması kitabxanasıdır. Bu kitabxana gözoxşayan və informativ statistik qrafiklər yaradmaq üçün istifadəsi və anlaşılması asan interfeys təqdim edir.

Seaborn'un bizə təklif etdiyi imkanlardan bəziləri:

1. Bir neçə dəyişən arasındakı əlaqələrə nəzər yetirmək üçün datasetə fokuslanmış API təklif edir
2. Ümumi statistikanı göstərmək üçün kateqorik dəyişənlərdən istifadə etmək
3. Tək dəyişənli və ya iki dəyişənli paylamaları vizuallaşdırmaq və bunları datanın alt çoxluqları arasında müqayisə etmək imkanı
4. Müxtəlif növ asılı dəyişənlər üçün xətti reqressiya modellərinin avtomatik hesablanması və vizuallaşdırılması imkanı
5. Mürəkkəb datasetlərin ümumi strukturuna uyğun vizuallaşdırma
6. Mürəkkəb vizuallaşdırmaları asanlıqla yaradmağa və oxumağa imkan verən birdən çox qrafiki grid -lər
7. Data -larınızdakı qanunauyğunluqları asanlıqla ortaya çıxarmaq üçün renk palitrasından seçmək üçün alətlər

Aşağıdakı koda ardıcıllıqla nəzər yetirək:

```
# seaborn
# https://seaborn.pydata.org/tutorial/introduction

# 1. seaborn və matplotlib.pyplot kitabxanasını import edirik
import seaborn as sns
import matplotlib.pyplot as plt

# defolt seaborn teamasını, miqyası və
# rəng palitrasını tətbiq edirik

sns.set()

# nümunə dataseti yükləyirik

tips = sns.load_dataset("tips")

sns.relplot(x="total_bill", y="tip", col = "time",
            hue = "smoker", style = "smoker", size = "size",
            data = tips)
plt.show()
```



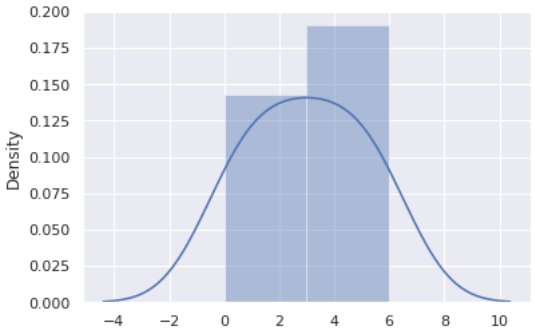
Kodları çalışdırdıqda yuxarıdakı qrafiklər görünər.

Bu xüsusi diaqramlar "**tips**" datasetindəki beş dəyişən arasındakı əlaqəni göstərir. Onlardan üçü ədədi, ikisi isə kateqorik dəyişənlərdir. İki ədədi dəyişən (total_bill və type) hər bir nöqtənin koordinat oxlardakı mövqeyini, üçüncü (size) isə hər bir nöqtənin ölçüsünü müəyyən edir. Kateqorik dəyişənlərdən biri dataseti iki fərqli qrafikə ayırır, digəri isə hər bir nöqtənin rəngini və formasını təyin edir.

Bütün bunlar seaborn kitabxanasının relplot() funksiyasına edilən tək bir müraciətlə reallaşır. Seaborn -un dataset dəyişənlərinin adlarını və plot(grafik) üzərində ifadə etməsini istədiyimiz rolları necə rahatlıqla təyin etdiyinə diqqət yetirək. Belə ki, burada birbaşa matplotlib -da istifadə etdiyimizdən fərqli olaraq, dəyişənləri parametrlərə çevirmək lazım deyil (məsələn, hər bir kateqoriya üçün istifadə ediləcək xüsusi rəng və ya marker). Bu seaborn tərəfindən avtomatik olaraq edilir. Bu, istifadəçiyə qrafikdə cavablandırılmasını istədikləri suala fokuslanmağa imkan verir. Seaborn kitabxanası haqqında daha ətraflı məlumat əldə etmək üçün keçid edin.

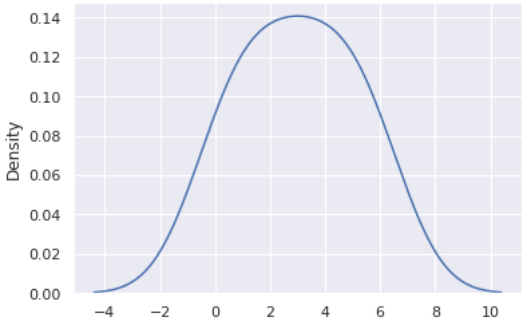
Seaborn, əlavə olaraq, random paylamaları vizuallaşdırmaq üçün də istifadə edilir.
import matplotlib.pyplot as plt ifadəsindən istifadə edilərək matplotlib modulunun pyplot obyekti koda import edilir.
import seaborn as sns ifadəsi ilə bu modul (seaborn) birlikdə istifadə edilə bilər.

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) #xəbərdarlığı söndürürük

sns.distplot([0,1,2,3,4,5,6])
plt.show()
```



Distplot paylama qrafikini ifadə edir, giriş kimi bir massiv alar və massivdəki nöqtələrin paylanmasına uyğun gələn əyri ((kilsə) zəng əyrisi) çəkir. Diaqramda histoqramların göstərilməməsi üçün hist = False xassəsini distplota əlavə etmək olar.

```
sns.distplot([0,1,2,3,4,5,6],hist=False)
plt.show()
```



Bu bölmədə mən qısa formada NumPy-də təsadüfi ədədlər, bu ədədlərin necə yaradıldığı, massivlərdə necə istifadə edildiyinin, verilənlərin təsadüfi paylanması, təsadüfi permutasiyaların necə yaradılacağı və NumPy-də statistik verilənlərin vizuallaşdırılması üçün istifadə olunan Seaborn kitabxanası haqqında danışdım.

Resurslar:

https://numpy.org/,
Seaborn: An introduction to seaborn

NumPy'da Random Sayılar, Random Veri Dağılımı ve Seaborn – Bölüm 10