

normalliktestleri

January 10, 2024

1 jarque bera test

Jarque-Bera testi, bir veri setinin normal dağılıma ne kadar uygun olduğunu sınamak için kullanılan bir istatistik testidir. Bu test, bir veri setinin çarpıklık (skewness) ve basıklık (kurtosis) özelliklerini temel alarak normal dağılıma ne kadar benzediğini değerlendirir.

[]: Bu test için hipotezler şöyle ifade edilir:

Sıfır hipotezi-H0: Veriler normal dağılım gösterir
Alternatif hipotez-H1: Veriler normal dağılım göstermez.

Test istatistiği örneklem basıklık ve çarpıklık ölçülerinin dönüşümlerinden $\sqrt{\frac{JB}{n}}$ elde edilmiştir.

$$JB = \frac{n-k}{6} \left(S^2 + \frac{(k-3)^2}{4} \right)$$

$$S = \frac{\mu_3}{\sigma^3} = \frac{\mu_3}{(\sigma^2)^{3/2}} = \frac{\frac{1}{n} \sum_{i=1}^n (x - \bar{x})^3}{\left(\frac{1}{n} \sum_{i=1}^n (x - \bar{x})^2 \right)^{3/2}}$$

$$K = \frac{\mu_4}{\sigma^4} = \frac{\mu_4}{(\sigma^2)^2} = \frac{\frac{1}{n} \sum_{i=1}^n (x - \bar{x})^4}{\left(\frac{1}{n} \sum_{i=1}^n (x - \bar{x})^2 \right)^2}$$

Eğer p-value (p-değeri) belirli bir anlamlılık düzeyinden küçükse (, genellikle 0.05 olarak seçilir), null hipotezi reddedilir. Yani, veri seti normal dağılıma sahip değildir. Eğer p-value belirli bir anlamlılık düzeyinden büyükse, null hipotezi reddedilemez. Bu durumda, veri setinin normal dağılıma sahip olduğu kabul edilir.

2 JARQUE BERA SİMÜLASYON

```
[288]: import numpy as np
from numpy import random
import pandas as pd
from scipy.stats import skew ,kurtosis
from scipy import stats
```

```

def datagenerator(n):
    # x = random.gamma(shape=2, scale=2, size=(1, n))
    x = random.normal(loc=0, scale=1, size=(1, n))
    # x = random.exponential( scale=2, size=(1, n))
    flattened_x = x.flatten()
    return flattened_x

def jarquebera(dataset):
    S = skew(dataset, axis=0)
    K = kurtosis(dataset, axis=0, fisher=True )
    n = len(dataset)
    jb = (n / 6) * (S ** 2 + (1 / 4) * (K - 3) ** 2)
    jb_pv = stats.chi2.sf(jb, 2)
    return jb_pv

def jarquebera_sim(n, simcount):
    truehypothesis=0
    falsehypothesis=0
    count = 0
    dataset=datagenerator(n)
    pval=jarquebera(dataset)
    print("jb olasılık değeri", pval)
    while (count<simcount):
        if pval<0.05:
            truehypothesis+=1
            count += 1
        else:
            falsehypothesis+=1
            count += 1

    return truehypothesis, falsehypothesis

```

```

[287]: simcount=100
        n=20

        sim_result1 = jarquebera_sim(100, 1000)
        print("H0 kabul eden simülasyon sayısı:", sim_result1[0])
        print("H0 reddeden simülasyon sayısı:", sim_result1[1])

```

```

6.234674008756455e-12
H0 kabul eden simülasyon sayısı: 1000
H0 reddeden simülasyon sayısı: 0

```

3 Cramér–von Mises test (one sample)

https://en.wikipedia.org/wiki/Cram%C3%A9r%E2%80%93von_Mises_criterion

konu anlatımı:<https://www.youtube.com/watch?v=pCz8WlKCJq8>

İstatistikte Cramér-von Mises kriteri, bir verisetinin, kümülatif dağılım fonksiyonu ile uyum iyiliğini değerlendirmek için kullanılan bir kriterdir. $F^{\{*\}}$ belirli bir ampirik dağılım fonksiyonu $F_{\{n\}}$ ile karşılaştırıldığında veya iki ampirik dağılımı karşılaştırmak için kullanılır.

Null Hipotezi (H_0): Örneklem dağılımı, belirli bir teorik dağılıma uyar.

Alternatif Hipotez (H_1): Örneklem dağılımı, belirli bir teorik dağılıma uymaz.

$x_{\{1\}}, x_{\{2\}}, \dots$ olsun. $x_{\{n\}}$ n artan sırada gözlenen değerler olsun. O zaman istatistik

$$T = n\varpi^2 = \frac{1}{12n} + \sum_{i=1}^n \left[\frac{2i-1}{2n} - F(x_i) \right]^2$$

Bu değer kullanılan tablodaki değerden büyükse, verilerin seçilen dağılımından geldiği hipotezi reddedilebilir.

```
[283]: import numpy as np
from scipy.stats import norm

def normal_datagenerator(n):
    # x = random.gamma(shape=2, scale=2, size=(1, n))
    x = random.normal(loc=0, scale=1, size=(1, n))
    # x = random.exponential( scale=2, size=(1, n))
    flattened_x = x.flatten()
    return flattened_x

def calculate_cramer_von_mises_sum_term(data):
    sorted_data = data

    # Number of observations
    n = len(sorted_data)

    # Calculate the cumulative distribution function (CDF) values for the data
    #cdf_values = [i / n for i in range(1, n + 1)]
    cdf_values = np.array([norm.cdf(x, loc=0, scale=1) for x in sorted_data])
    # Calculate the sum term
    sum=0
    for i in range(1, n + 1):
        sum += ((2 * i - 1) / (2 * n) - cdf_values[i - 1])**2
```

```

pop_std= np.std(data)
sum_term =(1/12*n) +sum
zstat=sum_term/pop_std
pvalue=2*(1-norm.cdf(zstat))
return pvalue

def cramer_von_mises_sim(n,simcount):
    truehypothesis=0
    falsehypothesis=0
    count = 0
    dataset=normal_datagenerator(n)
    pval= calculate_cramer_von_mises_sum_term(dataset)
    print("olasılık değeri",pval)
    while (count<simcount):
        if pval>0.05:
            truehypothesis+=1
            count += 1
        else:
            falsehypothesis+=1
            count += 1

    return truehypothesis,falsehypothesis

```

```

[285]: simcount=50
n=20

sim_resultt =cramer_von_mises_sim(n,simcount)
print("H0 kabul eden simülasyon sayısı:", sim_resultt[0])
print("H0 reddeden simülasyon sayısı:", sim_resultt[1])

```

```

1.4680540679989917e-05
H0 kabul eden simülasyon sayısı: 0
H0 reddeden simülasyon sayısı: 50

```

4 KOLMOGROV SMIRNOW TEST

KONU ANLATIMI:<https://www.youtube.com/watch?v=cltWQsmBg0k>

<https://www.statology.org/kolmogorov-smirnov-test-python/>

ksone distribution

https://docs.scipy.org/doc/scipy/tutorial/stats/continuous_ksone.html

<https://stackoverflow.com/questions/53509986/obtaining-the-critical-values-needed-for-the-kolmogorov-smirnov-test>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.smirnov.html>

Bu test de bir uyum iyiliği testidir.(goodness of fit)Testin amacı gözlenen frekanslar ile beklenen frekansların birbirine ne düzeyde benzeştiğine dayanır.

Null Hipotezi (H0): $F_0(X)=S_N(X)$ ————— $D_h < D_{\alpha,n}$

Alternatif Hipotez (H1): $F_0(X) \neq S_N(X)$ ————— $D_h > D_{\alpha,n}$

Test istatistiği, örneklem veri setinin kümülatif dağılım fonksiyonu ile belirli bir teorik dağılımın kümülatif dağılım fonksiyonu arasındaki en büyük farkı temsil eder.

$$D = \max[F_0(x) - S_n(x)] \quad S_N = k/N$$

Şeklinde hesaplanır ve Tablo değerleri ise ksone dağılımı (1- smirnow dağılımı) üzerinden hesaplanır.

```
[279]: import numpy as np
from scipy.stats import norm
from numpy import random
from scipy.stats import ksone

def ks_critical_value(n_trials, alpha):
    return ksone.ppf(1 - alpha / 2, n_trials)

def ks_datagenerator(n):
    # x = random.gamma(shape=2, scale=2, size=(1, n))
    x = random.normal(loc=0, scale=1, size=(1, n))
    flattened_x = x.flatten()
    return flattened_x

def calculate_kstest(data):
    n = len(data)
    cdf_values = [i / n for i in range(1, n + 1)]
    normal_values = np.array([norm.cdf(x, loc=0, scale=1) for x in data])
    sup = [abs(cdf_values[i - 1] - normal_values[i - 1]) for i in range(1, n + 1)]
    supremum_value = max(sup)
    return supremum_value

def ks_sim(n, simcount):

    truehypothesis = 0
    falsehypothesis = 0
    count = 0
    dataset = ks_datagenerator(n)
    critical_value = ks_critical_value(n, 0.05)
    pval = calculate_kstest(dataset)
    print("olasılık değeri", pval)
    print("kritik değeri", critical_value)
    while count < simcount:
```

```

    if pval < critical_value:
        truehypothesis += 1
        count += 1
    else:
        falsehypothesis += 1
        count += 1

    return truehypothesis, falsehypothesis

```

```

[278]: sim_result = ks_sim(100, 1000)
print("H0 kabul eden simülasyon sayısı:", sim_result[0])
print("H0 reddeden simülasyon sayısı:", sim_result[1])

```

```

0.9191298846876634
0.13402815758236203
H0 kabul eden simülasyon sayısı: 0
H0 reddeden simülasyon sayısı: 1000

```

4.1 kontrol için 4.soru

[https://acikders.ankara.edu.tr/pluginfile.php/129673/mod_resource/content/0/Veri%20Analizi%207.%20Hafta.p](https://acikders.ankara.edu.tr/pluginfile.php/129673/mod_resource/content/0/Veri%20Analizi%207.%20Hafta.ppt)

```

[68]: F0 = [0.015, 0.0425, 0.06, 0.095, 0.13, 0.1675, 0.2325, 0.24, 0.3525, 0.3875, 0.
    ↪ 4025, 0.45, 0.575, 0.6475, 0.65, 0.73, 0.78, 0.82, 0.94, 0.955]
SN = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.
    ↪ 7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0]
n = len(F0)
sup = [abs(F0[i - 1] - SN[i - 1]) for i in range(1, n + 1)]
supremum_value = max(sup)
critical_value = ks_critical_value(n, 0.05)
print(supremum_value)
print(critical_value)
print("sorudaki veri ile karşılaştırdığımızda sonuçların doğru olduğunu,
    ↪ görebiliyoruz.")

```

```

0.16000000000000003
0.2940755433823519

```

5 KAYNAKÇA

1. A comparison of various tests of normality - Berna Yazici, Senay Yolacan
2. https://acikders.ankara.edu.tr/pluginfile.php/42377/mod_resource/content/0/2.%20HAFTA.pdf