

Statistics Labs

Subject :

Date :/.....

(*) import seaborn as sns

(*) sns.get_dataset_names() (Dataset isimlerini öğrenmek için kullanılır)

(*) df = sns.load_dataset("mpg")

dataframe

(*) df.head() # İlk 5 satır getir.

(*) df.shape # Kaç satır kaç sütun var.

(*) df.columns # Column isimlerini verir.

(*) len(df.columns) # Column sayısını verir.

(*) df.info() # Integer olanlar arasında da kategorik olabilir. Mesela "model year" kategorik olarak değerlendirilebilir.

(*) df.select_dtypes(include=["object"]) # dtype'i sadece object olanları çıkarır. Bir filtreleme methodu.

(*) df.nunique(axis=0) # Her bir variable'ın kaç tane unique değer var onu gösterir

- (*) df.plot.pie(y = "cylinders") # Dairevi silindirlik bir grafik yaptı, tem anlaşılmı.
- (*) df.describe() # Kategorikleri almadı. Sadece numeric standarı gösterdi.

Population And Sample

(*) import numpy as np

Aynı random sayıları üretmek için seed methodu kullanılır
ve parantez içinde aynı sayı yazılır.

(*) np.random.seed(101)

population = np.random.randint(0, 100, size=100000)

1-100 arası rastgele sayılar oluşturduk

(*) len(population)

(*) np.random.seed(51)

sample = np.random.choice(population, 100)

Hacaille aynı rastgele sayılar oluştur. Çünkü seed methodu kullanıp aynı sayıları kullandık.

(*) population.mean()

Population'ın ort. 'ını aldı. (Şeçim sonucu örnek verilebilir.)

(*) sample.mean()

Sample'ın ort. 'ını aldı. Popülasyona yakın bir ort. çıktı.
(Şeçim tahminlerin örnek verilebilir.)

100 tane sample'ın ort. 'ını bulmak potiyant diyelim:

```
sample_means = []
```

```
for i in range(100):
```

```
    sample = np.random.choice(population, 100)
```

```
    sample_mean.append(sample.mean())
```

```
sample_mean
```

Her seferinde rastgele seçilen 100 kişil 'ı ayrı olmaysın
için her seferinde farklı sonucu beklemeli ort. değer,
popülasyon ort. 'ıdır. Yani bu da 50'dir.

Sample'ın ort. 'ı da population ort. 'ına yakın çıktı.
Sample bittiğimdeki gibi, Sample size', 'kesinlikle
daha sağlam sonucu elde ettiğimiz gibi çok.

(*) np.mean(sample_mean)

→ 48.9116

Import numpy as np

Import matplotlib.pyplot as plt

Import seaborn as sns

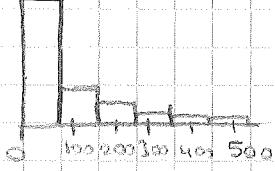
from scipy import stats

Titanic datasetini inceleyiniz.

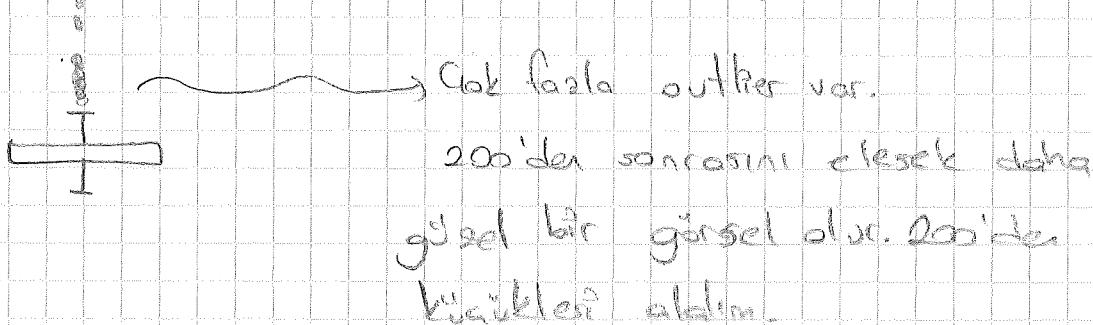
df["fare"] veya df.fare # Fare suyunu incelemek istiyorum.

(*) stats.skew(fare) # Pozitif bir skew var.
4.779

(*) plt.hist(fare); # Bir grafik oluştur.
Std yıkık
Light skew
Outlier var.



(*) plt.boxplot(fare); # Outlier tespiti için kullanılır.



fare <= 200

True, True, --
True, --, --

\Rightarrow Boolean degeri oluştur.

Subject :

Date

$$\text{boolarr} = \text{fare} <= 200$$

$$\text{new fare} = \text{fare}[\text{boolarr}]$$

(*) np.std(new fare)

$$\rightarrow 29.12129$$

(*) np.percentile(new fare, 25)

$$\rightarrow 7.8958$$

(*) np.percentile(new fare, 50)

$$\rightarrow 13.8625$$

(*) np.percentile(new fare, 75)

$$\rightarrow 30.0$$

(*) iqr = np.percentile(new fare, 75) - np.percentile(new fare, 25)

$$195$$

→

(*) stats.iqr(new fare)

Subject:

Date:

Scatter Plot

2 adet değişkenin olusarı veri setlerinde kullanılsı.

$tv_hours = [3, 5, 2, 0.5, 3, 1, 4, 3, 4]$ # Öğrencilerin TV izleneceği saatler?

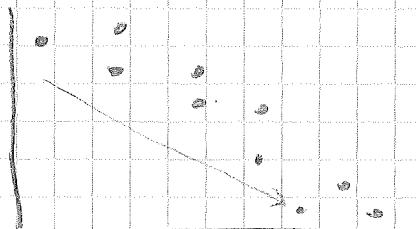
GPA = [2.3, 2.1, 3.3, 3.4, 2.0, 3.0, 3.6, 2.8, 3.5, 2.6] # notları.

$tv_hours = np.array(tv_hours)$

GPA = np.array(GPA)

(*) $tv_hours * 3$

(*) $plt.scatter(tv_hours, GPA);$ # Görüel olarak aralarında ilişkili görülmek istedik.



► Bir değişkenin kendisyle covariance'sı birbirinden farklı bir rakam elde edilirken, correlation'ı 1 civar.

$np.corrcoef(tv_hours, GPA) \rightarrow$ 2 boyutlu bir array var.

→ array([[-1., -0.85697808], [-0.85697808, 1.]])

TV'nin kendisyle korelasyonu

GPA'nın kendisyle korelasyonu

TV ve GPA'ının birbirileyle korelasyonları

1 milyon kere 2 zar atılık

Subject:

05.11.2021

Büyük sayılar kanunu

Date:

$$n = 1000000$$

→ Ne kadar artarsa göreceli frekans
gerçek frekansa yaklaşırlı

(*) `np.random.seed(51)`

$die_1 = np.random.randint(1, 7, size=n) \rightarrow$ İlk zar

(*) `np.random.seed(81)`

$die_2 = np.random.randint(1, 7, size=n)$

$$X = die_1 + die_2$$

`df = pd.DataFrame([{"die_1": die_1, "die_2": die_2, "X": X}])`

`df.head(10)`

$total_four = df[df.X == 4] \rightarrow X'in 4 olma
durumları$
`len(total_four)`

$p = len(total_four) / len(df) \rightarrow$ Görelebilir frekans
($total_four/n$)

$$1/12$$

→ Teorik frekans → Yaklaşım. br.
(Büyük sayılar
kanunu)

(*) `freq = df.X.value_counts()`

`freq`

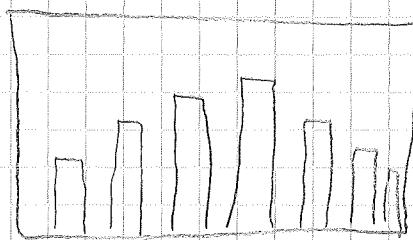
→ Hangi toplandan kaç tane
üretildi mis?

Subject :

Date :

%matplotlib inline

freq.sort_index().plot(kind="bar", grid=True);

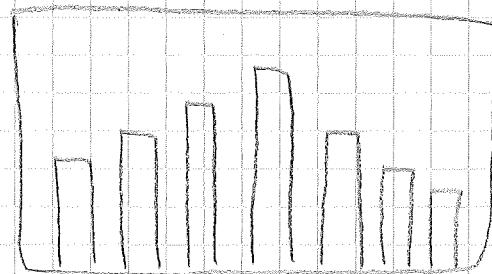


prob = df.x.value_counts() / n

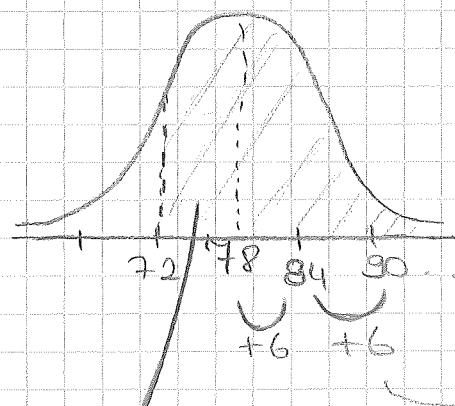
prob → Görecek frekanslar

%matplotlib inline

prob.sort_index().plot(kind="bar", grid=True);



Qo Bir sınırlı dağılım, ort = 78, var = 36
Bu sınırlıda bir kişiin 42' den yüksek olma olasılığı?



$$\text{Var} = 36 \quad \text{ise}$$

$$\text{std} = \sqrt{36} = 6$$

Bize 42' den büyük kisimın alanını soruyor.

$$\mu = 78 \quad (\text{ort})$$

$$\sigma = np \cdot \sqrt{36} \quad (\text{std})$$

$$x = 42 \quad (\text{random variable})$$

Olasılık hesabı:

`stats.norm.cdf(x, mu, sigma)` \Rightarrow Bu sonuc 42'ye kadar olan olasılık

42' den sonraki olasılık $= 1 - \text{cdf}(42)$

$$1 - \text{stats.norm.cdf}(x, \mu, \sigma)$$

ppf(q, loc=0, scale=1)

cdf'in tersi

cumulative bir
değer olacak mean

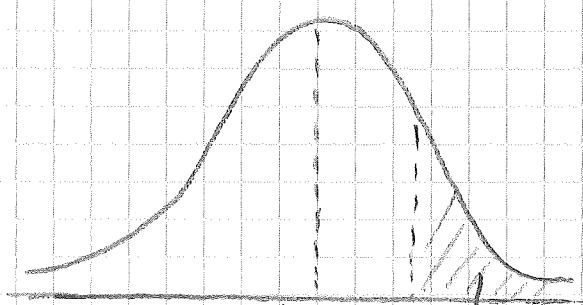
std

Subject:

Date:

b) Sınıfin top %10'lu A notunu alıyor.

A notunu alabilmek için min skor kaç olmalı?



78

→ %10'luk bir kısım istiyor.

! X değeri verildiğinde, X'den olasılığa giderken

cdf kullanıyoruz. Olasılıktan X'e giderken ise

ppf methodu kullanılır.

ppf(q, loc=0, scale=1) formülünden:

$$q = 1 - 0.1 = 0.9 \quad (\%10'dan geriye kalanlar \%90'luk kısmın
alanı için 1'den çıkardık)$$

$$\mu = 78$$

$$\sigma = np \cdot \sqrt{36}$$

Sırayla yazarsak mu ve
scale yazmak zorunda
değiliz

stats.norm.ppf(q, loc=μ, scale=σ)

Output → 85.68



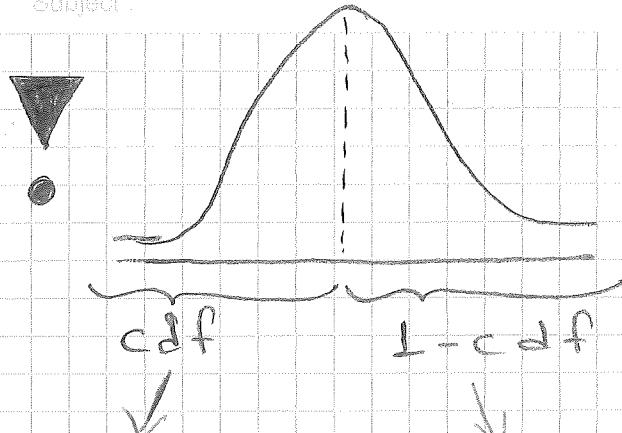
Yani bu puan ve üstü alanlar
%10'luk kısma girmis.

X' den olasılık hesabı için

cdf

Subject:

Date:

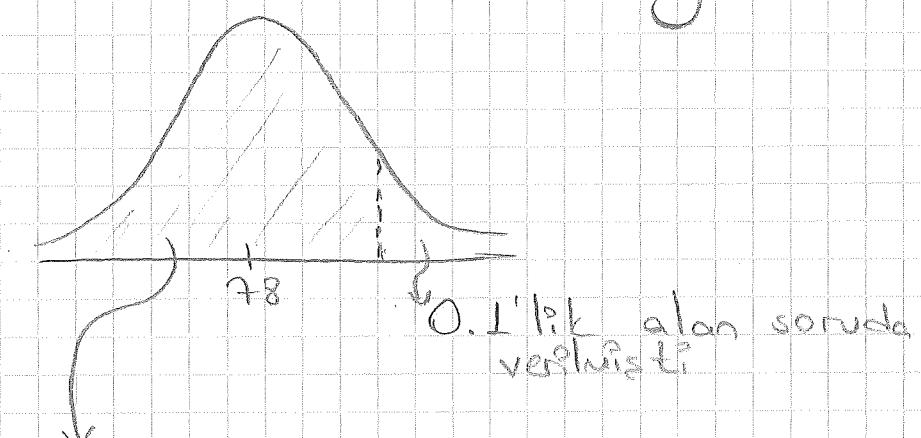


Sol taraf
için cdf

Sağ taraf için
1 - cdf

! Olasılık hesabından X' e gitmek için ppf

ppf

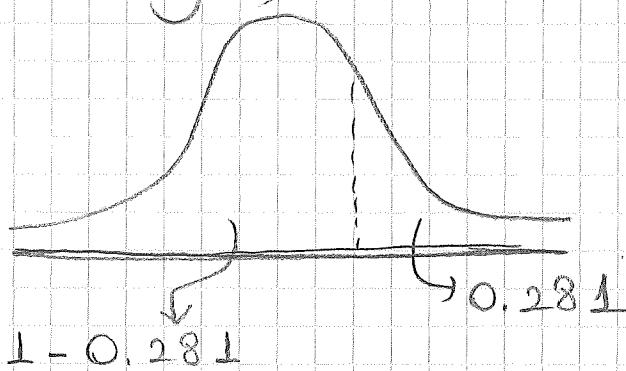


Göre kalan alan

0.9 olur. Yani $q = 0.9$

Top % 28.1' e karşılık gelen değer nedir?

(Önceki sonunun aynı olasılık verilisi, ordan X' e gitileceğiz. O zaman ppf methodu kullanacağız)



$$q = 1 - 0.281 = 0.719$$

$$\mu = 78$$

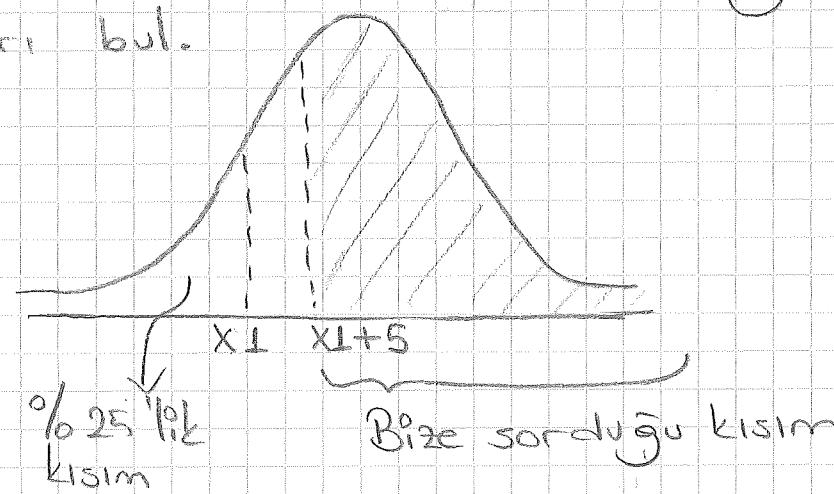
$$\sigma = np \cdot \text{sqrt}(36)$$

`stats.norm.ppf(q, mu, sigma)`

Output $\Rightarrow 81.47$

Yani bir kişinin top % 28.1'lik diline girmesi için 81.47 veya 82' not alması gerekiyor.

- (d) En soldaki $\%25$ 'lik kisim kesen cuts off point'i bul; bu naktanın 5 veya fazlası not alanları bul.



$$q_L = 0.25 \quad (\text{Olasılık})$$

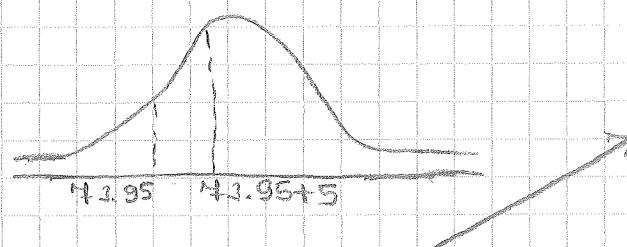
$$X_L = \text{stats.norm.ppf}(q_L, \mu, \sigma)$$

Olasılık $\%25$ olarak verildiği için olasılıktan X 'e gitmem gerekiyor.
ppf yöntemi kullanılır.

Output : 73.95

$$\hookrightarrow \text{Yani } X_L = 73.95$$

Ama benden bu naktanın 5 fazlasını istedim:



X 'den olasılığı gitmem için
cdf yöntemi

$\text{stats.norm.cdf}(X_L+5, \mu, \sigma) \Rightarrow$ Sol tarafın
alanı

$1 - \text{stats.norm.cdf}(X_L+5, \mu, \sigma) \Rightarrow$ Sağ tarafın
alanı

t - Distribution

t dağılımı çizileceğinde sorulması gereken :

"What is degrees of freedom?" = (df)
(Serbestlik derecesi)

$$\text{Serbestlik derecesi} = n - 1$$

! 30' dan büyük sample' larda populasyonun standart sapması bilinmiyorsa t - dağılımı kullanılır.

t - dağılımları normal dağılıma yakındır ama tailed larda biraz sisman olurlar.

`stats.t.ppf(q, df, loc=0, scale=1)`

↓
cumulative
olasılık

↓
serbestlik
derecesi

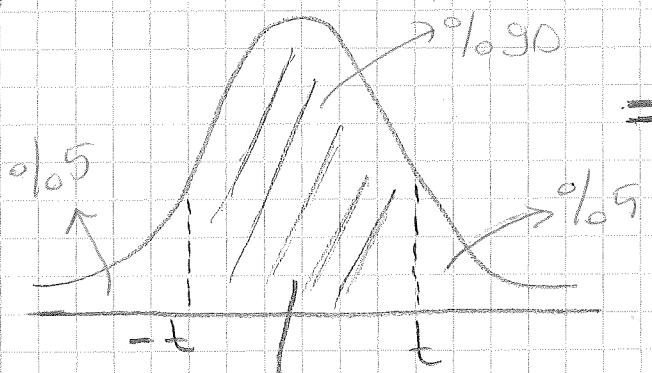
↓
mean

↓
std

! 30' dan büyük sample' larda 2 dağılımı
(Normal distribution)



Serbestlik derecesi 15 olan bir t. dağılımı
vat. Dağılımin $\% 90$ 'ı, ortalamanın kaç standart
sapması içindedir?



\Rightarrow Bütün olıboren t'ye
kadar gelebilmem için
 $\% 90 + \% 5 = \% 95$ lik
bir alan bulmamız.

$\% 90$ 'lik kısım
icerde kalsın
istiyoruz. O zaman z değer kaçır?

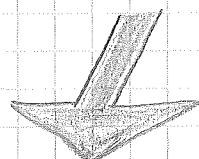
Olasılıktan random variable'ı giàyıyoruz ppf yöntemi
kullanacağız.

$$df = 15$$

$$q = 0.95$$

$$\text{stats.t.ppf}(q, df)$$

Output $\Rightarrow 1.75$



\Rightarrow Standart bir t dağılımı-
nın olduğu için mean ve
std 'yi kullanmadık. Kendisi
default olarak yazdı.

1.75 standart sapma sapa,

1.75 standart sapma sola gitmişinde

dağılıminin $\% 90$ 'ı içerde kalsın



- t dağılımında kuyruklar data sisman olduğu için
- ortaya kalan alan data kwasık çıktı.

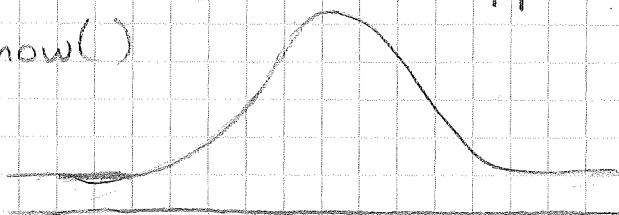
GRAFIKLER

$df=1$ olan bir t dağılımı çizdirelim:

```
plt.figure(figsize=(8, 6))
```

```
xS = np.linspace(-5, 5, 100)
```

```
plt.plot(xs, stats.t.ppf(xs, 1), "b", label="t(df=1)")  
plt.show()
```



\rightarrow $df=3$ \rightarrow \leftarrow
 \rightarrow $df=3$ \rightarrow \leftarrow
 \rightarrow $df=3$ \rightarrow \leftarrow
 \rightarrow $df=3$ \rightarrow \leftarrow

```
plt.plot(xs, stats.t.ppf(xs, 3), "g", label="t(df=3)")
```

\downarrow
 \rightarrow $df=3$ \rightarrow \leftarrow
 \rightarrow $df=3$ \rightarrow \leftarrow
 \rightarrow $df=3$ \rightarrow \leftarrow

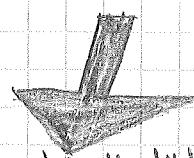
```
plt.plot(xs, stats.t.ppf(xs, 8), "m", label="t(df=8)")
```

\downarrow
 \rightarrow $df=15$ \rightarrow \leftarrow
 \rightarrow $df=15$ \rightarrow \leftarrow

```
plt.plot(xs, stats.t.ppf(xs, 15), "k", label="Normal")
```

```
plt.legend()
```

```
plt.show()
```



df boyadikce 'tail' de sismanlik
azalir, orta kisim yukari dogru cikar.

Normal dagilima yaklasir. Yani sample
sayisi artikca normal dagilima yaklasir.

30'un istenide olap sample size'larda
normal dagilim kullanabiliriz.

Confidence Intervals Using the Normal Distribution

import seaborn as sns

sns.get_dataset_names() → Seaborn'dan dataset isimlerini aldık.

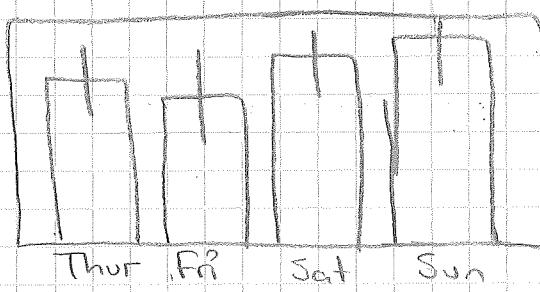
Biz tips'i kullanacağız



tips = sns.load_dataset("tips")

① "total_bill" Üzerine vertical barplot olustur. "day" ile de grupta.

sns.barplot(x="day", y="total_bill", data=tips, ci=95)
plt.show()



↳ given analysis

↳ Genelik total_bill'leri aldık
Üzerine de confidence interval'ları da aldık

- ② Subset a dataframe just including Sunday tips.

`tipsSun = tips[tips.day == "Sun"]`

↳ Tablodan sadece Sun satırlarını seçtiğimizde.

- ③ Mean ve std'yi bul.

`tipsSun.total_bill.mean()` → total_bill üzerinde çalışıyoruz.

`tipsSun.total_bill.std()`

- ④ Calculate standard error of the mean

$$\text{Standard Error} = \frac{\text{std}}{\sqrt{n}} = \frac{\text{tipsSun.total_bill.std()}}{\text{np.sqrt(len(tipsSun))}}$$

$$\text{sem} = \text{tipsSun.total_bill.std()}/$$

$$\text{np.sqrt(len(tipsSun))})$$

Populasyonda $\rightarrow \frac{\sigma}{\sqrt{n}}$

Sample'da
(Dataset'te) $\rightarrow \frac{s.d.}{\sqrt{n}}$

(5) Calculate the margin of error (%95)

$$\boxed{moe = 2 \cdot \frac{std}{\sqrt{n}}}$$

$$z = \text{stats.norm.ppf}(0.975)$$

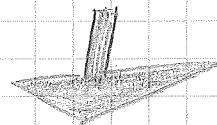


Olasılıkta random variable'a giderek z 'yi bulduk.

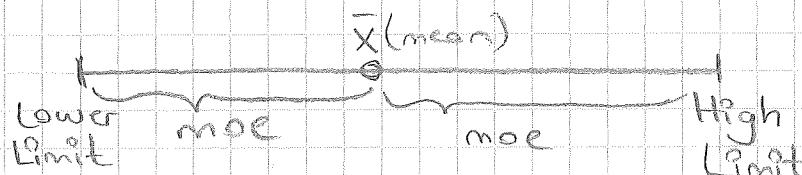
$$moe = 2 \cdot sem$$

$$moe = 2 \cdot \frac{std}{\sqrt{n}}$$

$$\text{output} \Rightarrow 1.98$$



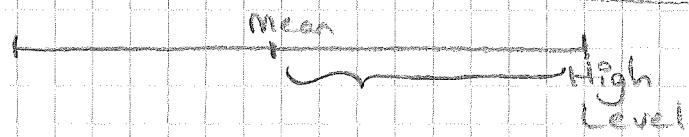
Ortalamanın "uzerine" %95'lik bir güven aralığı: $\bar{x} \pm moe$
ettim. Hata margin'i 1.98



⑥ Calculate the upper confidence limit.

mean'i alip 'yaşine hata margin'i, kayarSAN
high level'i, bulurum.

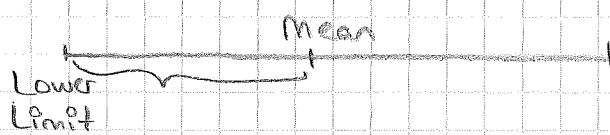
`tipsSun.total_bill.mean() + moe`



⑦ Calculate the lower confidence limit

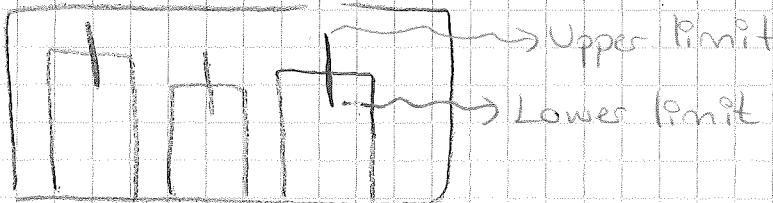
mean'i den hata margin'inin aksarısak lower
limit'i buluruz.

`tipsSun.total_bill.mean() - moe`



▼ Bulduğumuz lower-high limit, barplot'daki

ci'nin üst ve alt değerleri



(8) Calculate 95% Confidence Interval around the mean by using `scipy.stats`

`stats.norm.interval(alpha, loc=0, scale=1)`

↓
 confidence
interval (ci)
 ↓
 mean
 ↓
 std error
of the
mean

`stats.norm.interval(0.95, loc=tipsSun.total_bill.mean(),
scale=sem)`

↓
 given
aralığı (ci)
 ↓
 mean
 ↓
 std error

output (19.42, 23.39)

↓
 Lower
Limit
 ↓
 High
Limit

Onceki 2 sonucu aynı aynı
bulmuştuk. Burda bir kodla %95
değeri aynı anda verdi.

Değerlerini bilmemiş olduğumuz bir datasetten Sunday'i çekerek
bir veriseti oluşturduk. Mean'i bulduk ama yetersiz. Bunun için
bir güven aralığı istedik, %95'lik bir güven aralığı tanı
ettik. Bu doğrumsuz aralık 19.4 ile 23.4. Popülasyonun gerçek
ortalaması %95 şanslı ile bu ikinci değer arasında.

Exercise

64 müşterinin alışveriş süresi kaydedilmiştir.

Average $\bar{x} = 33$ minutes

Variance = 256 minutes²

$1 - \alpha = .90$ $M = ?$

→ %90 güvenglilikte ortalamanın denk geldiği
alan hesabını yapınız.

$n = 64$

$\bar{x}_{\text{bar}} = 33$

var = 256

$\alpha = 0.90$ (Confidence level)

$s = \sqrt{n} \cdot \text{sqrt(var)}$ (standart sapma)

$$\text{sem} = \frac{\text{std}}{\sqrt{n}} \Rightarrow \text{sem} = s / \sqrt{n} \cdot \text{sqrt}(n)$$

Standart hata

Güven aralığını hesaplayacağız. Örnek sayıımız 30'dan büyük
olduğu için büyük bir sample. Normal dağılım kullanabiliriz.

stats.norm.interval(0.90, loc = \bar{x}_{bar} , scale = sem)

give
aralığı

mean

std error

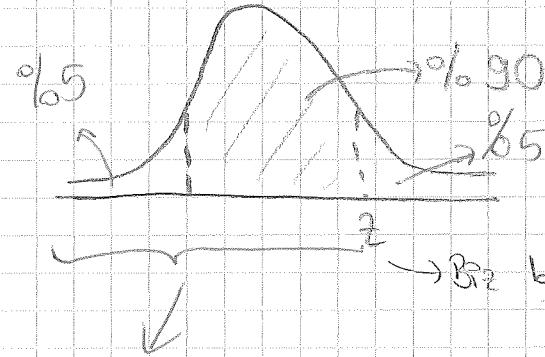
Output $\Rightarrow (29.71, 36.28)$

↳ Benim gerçek ortalamam $\%90$ ihtimal ile
bu aralikta.

Proof \Rightarrow

$$\bar{X} + \frac{2}{\sqrt{n}} S \xrightarrow{\text{moe}}$$

%90 olasılıktan 2 değerine geçiş yapmak
yapmak için ppf kullanmanız lazımlı.



\rightarrow Bu 2 değeri istiyoruz.

Cumulative olasılığımız $\Rightarrow %90 + %5 = %95$

$$z = \text{stats.norm}(0.95)$$

\rightarrow Norm dağılım olduğu için
loc ve scale default.

$$\text{moe} = 2 \cdot \text{sem}$$

$$\text{low_limit} = \bar{X} - \text{moe}$$

↓
(mean)

$$\text{high_limit} = \bar{X} + \text{moe}$$

Exercise

Yeni bir barışt öğretim firması var 8 mermiyi test ediyor ve namlu çıkış hızlarını ölçüyor:

3005 2925 2935 2965

2995 3005 2937 2905

Populasyon ortalaması hesabi təhnisi için $\alpha = 0.05$ 'lik güven aralığını bulunuz.

Küçük bir sample olduğunu için t -distribusyonunu kullanacağız.
 t -distribusyonunda okullarda serbestlik derecesi gelir.

$$\text{Serbestlik derecesi} = n - 1 = df$$

$$X = [3005, 2925, 2935, 2965, 2995, 3005, 2937, 2905]$$

$$n = 8 \quad (\text{sample size})$$

$$\bar{x} = np.\text{mean}(x) \quad (\text{sample mean})$$

$$s = np.\text{std}(X, ddof=1)$$

$$cl = 0.95 \quad (\text{confidence level})$$

! s = np.std(x) dersim
 population std 'yi hesaplar.

$ddof=0 \rightarrow \text{population std}$

$ddof=1 \rightarrow \text{sample std}$

`stats.t.interval(0.95, df=n-1, loc=xbar,`

`scale=s/np.sqrt(n))`

Subject:

Lab 2

8/1

Proof $\Rightarrow \bar{X} \pm t \frac{s}{\sqrt{n}}$

$$t = \text{stat. } t \cdot \text{ppf}(0.975, df=n-1)$$

\downarrow
Olasılığım biliyorum. t hesabı yapacağım. O zaman
ppf

$$\text{sem} = s / \sqrt{np} \cdot \sqrt{n}$$

$$\text{moe} = t \cdot \text{sem}$$

$$\text{low_limit} = \bar{X} - \text{moe}$$

$$\text{high_limit} = \bar{X} + \text{moe}$$

Subject :

16.11.2021

Date :

One Sample T-Test

According to Reynolds Intellectual Ability Scales, the average VIQ (Verbal IQ scores based on the four Wechsler (1981) subtests) is about 109.

In our sample data, we have a sample of 40 cases.

Let's test if the average VIQ of people is significantly bigger than 109.

Entellektüel yetenek Ölçegi'ne göre jözel IQ həsabı
mean = 109.

n = 40.

109 olarək kəbul edilən art. bür slor aseba,
109'dan böyük olubdur mii?

Tek grupp olduğunu rəqəm "One sample t-Test"

```
df=pd.read_csv("brain-size.csv", sep=";")  
na_values=".", index_col=0)
```

① Assumptions :

n = 40

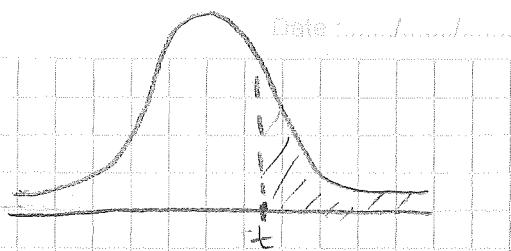
mean = 109

One sample t-Test

② Hypothesis:

$$H_0: \mu = 109$$

$$H_a: \mu > 109 \rightarrow \text{Right tail test}$$



③ Test Statistic:

$$\bar{x} = \text{df.VIG.mean}() \rightarrow \text{Ort. hesapladık}$$

$$s = \text{df.VIG.std}() \rightarrow \text{Std 'yi hesapladık}$$

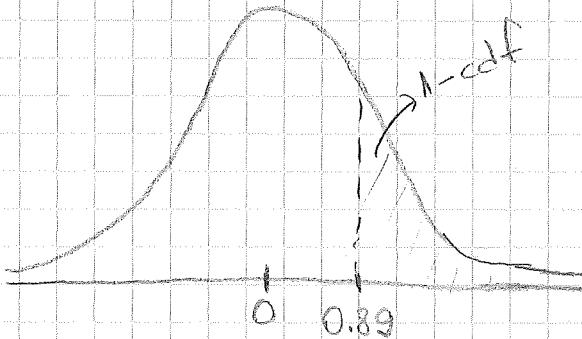
$$t\text{-test} = \frac{(\bar{x} - \text{mean})}{(s / \sqrt{\text{df.shape}[0]}))} = 0.89$$

④ P-value:

$$1 - \text{stats.t.cdf}(t\text{-test}, 39) \rightarrow \text{df}$$

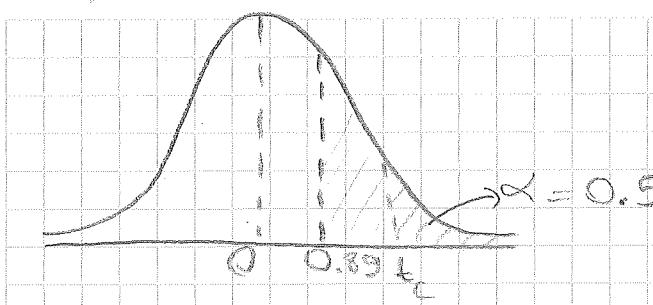
output: 0.187

⑤



$$\begin{aligned} \text{P-value} &> \alpha \\ 0.187 &> 0.05 \end{aligned}$$

H_0 hipotezinin redildeğinkinden ekreme yeterli konut yok. "fail to reject"



$P > \alpha$

veyse

$t_c < t_{kritik}$

de bulşak

ayni varsa

olarak

$(t_c = t_{kritik})$

► One Sample t-test (Sayı)

`stats.ttest_1samp(a, mean, alternative="two-sided")`

array-like \rightarrow sayılar
başlangıç \rightarrow başlangıç
ortalama \rightarrow ortalaması

`oneSamp = stats.ttest_1samp(df.VIQ, 109,`

`oneSamp` \rightarrow alternatif = "greater"

`oneSamp.pvalue = ,`

output: 0.184

Tek tane
test yapılık

`alpha = 0.05`

`if oneSamp.pvalue < alpha:`

`print("Reject the null")`

`else:`

`print("Fail to reject")`

! Varyanslar eşit mi değil mi önce onun hipotezi
yapılır. Sonra göre "equal" veya "not equal" formülü
Subject: \rightarrow kullanılır.

Date: 11.11.2023

Independent Samples T-Tests

Arsenic Example:

Arsenic concentration in public drinking water supplies is a potential health risk.

An article in the Arizona Republic (May 27, 2001) reported drinking water arsenic concentrations in parts per billion (ppb) for 10 metropolitan Phoenix communities and 10 communities in rural Arizona.

You can find the data in CSV file.

Determine if there is any difference in mean arsenic concentrations between metropolitan Phoenix communities in rural Arizona.

Şehir içme sularında arsilik konsantrasyonuyla ilgili bir sağlık riski varmış. Arizona Republic'de bir makale yayımlanmış. Bu makalede, metropolitan için 10 örnek, Phoenix için 10 örnek arsenik konsantrasyon değerleri var.

Bu riski community ort. arasında bir fark var mı?

① Assumptions:

Date:

arsenic = pd.read_csv("arsenic.csv")
arsenic.head()

Metro Phoenix	X1	Rural Arizona	X2
- - -	- -	- - -	- -
- - -	- -	- -	- -
- - -	- -	- - -	- -
- - -	- -	- - -	- -

10 sample var. t Test uygulanacak

! Hypothesis: → Hangi testi kullanacagimiz hipotesi?
(Varyanslar esit mi değil mi?)

$$H_0: \mu_1 = \mu_2$$

$$H_1: \mu_1 \neq \mu_2$$

Perform Levene test for equal variances.

The small p-value suggests that the populations do not have equal variances

→ İki farklı grubu bir önekleme birbirine esit varyansı mı var yoksa farklı mı?"

Buna "Esit Varyanslar Testi"

"Varyansların Homojenliği Testi"

"Levene Test"

Daha önce deste
yazdim.

deniyor.

Bu da "iki populasyonun varyansı esittir" der.

O zaman varyansların esit olmadığı formül kullanılır.

Levene Test

Levene Test Kullanarak önce eşit varyanslı mı? Düşünçimiz var mı yok mu öğrenelim.

Levene Test'in hipotezleri:

H_0 : The population variances are equal $\rightarrow X_1 = X_2$ (Eşitlik Sırasına)

H_1 : There is a difference between the variances in the population. $\rightarrow X_1 \neq X_2$

The small p-value suggests that the populations do not have equal variances.

leveneTest = stats.levene(X_1 , X_2)

↳ array-like

leveneTest = stats.levene(arsenic.X₁, arsenic.X₂)

output: statistic: 7.4015, pvalue = 0.012



P-value oldukça küçük çıktı.

Reject the null deniz.

! Fakat bu, Levene testinin
• p değeri

H_1 'i kabul ediyoruz.

Yani $X_1 \neq X_2$



Not equal formülini kullanacağız.

mean 1 = arsenic.X1.mean()

mean 2 = arsenic.X2.mean()

② Hypothesis:

$$H_0: \mu_1 = \mu_2$$

$$H_1: \mu_1 \neq \mu_2$$

③ Test Statistic:

ind-test = stats.ttest_ind(arsenic.X1, arsenic.X2,
 output: statistic=-2.76, pvalue=0.015) } equal_var=False)

Not equal var
formülüm
Kullalanın
gerektirsin
TtestTest'te
antlaşma.

alternative
yazınca şebe
yok, çıktı
default
two-sided

ind-test.statistic } Ayrı ayrı yardım dök
 output: -2.76 }

ind-test.pvalue

output: 0.015 }

↓ pvalue oldukça küçük çıktı

5 Conclusion :

$\alpha = 0.05$ (Default)

if `indTest.pvalue < alpha`:

print ("Reject the null")

else:

print ("Fail to reject the null")

Output : Reject the null.

" $\alpha = 0.05$ anlamılık seviyesinde (%95 güvenirlikte) 2 bölgein ortalamaları arasında istatistiksel anlamlı bir fark vardır."

Veya

"There is a statistical significant difference between two communities."



Bağımsız testlerde önce varyansların eşit olup olmadığına bakılacak, eğer "iki grup arasındaki varyans eşittir." hükmüne varırsak : (Levene testi ile)

`stats.ttest_ind(x1, x2, equal_var=True)`

"Varyanslar eşit deildir." hükmüne varırsak :

`stats.ttest_ind(x1, x2, equal_var=False)`

Pre-test Post-test

Subject:



Date:

Paired (Dependent) Sample T-test

Prozac Data

Let us consider a simple example of what is often termed "pre/post" data or "pretest/posttest" data.

Suppose you wish to test the effect of Prozac on the well-being of depressed individuals, using a standardised "well-being scale" that sums Likert-type items to obtain a score that could range from 0 to 20.

Higher scores indicate greater well-being (That is, Prozac is having a positive effect.)

While there are flaws in this design (e.g., lack of a control group) it will serve as an example of how to analyse such data.

Determine if Prozac enhances well-being in depressed individuals. Use 0.05

Likert-Type bir "özellikle" "well-being scale" "özelliklen-
dirmesi" yapılıus.
0 - 20 arası
(deprest) (deprest değil)

Depresif bireylere Prozac testi mi uygulanmış. Önceki ve
sonraki depresifler arasında farklı var mı?

① Assumptions:

Date:

prozac = pd.read_csv ("prozac.csv")

prozac

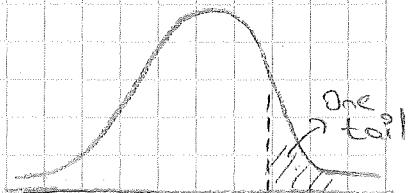
moodpre	moodpost	difference
- - -	- - -	- - -
- - -	- - -	- - -
- - -	- - -	- - -
- - -	- - -	- - -

↳ d-bar (farkların ort. ̄sı)

② Hypothesis:

H0: d-bar = 0 (fark yok)

H1: d-bar > 0 (Fark var)



↙ moodpost - moodpre farkının artması, iyileşme oldugu antamına gelir.

③ Test Statistic:

Dependent
+ test



stats.ttest_rel(x1, x2, alternative='two-sided')

rel-test=stats.ttest_rel(prozac.moodpost, prozac.moodpre,
alternative="greater")

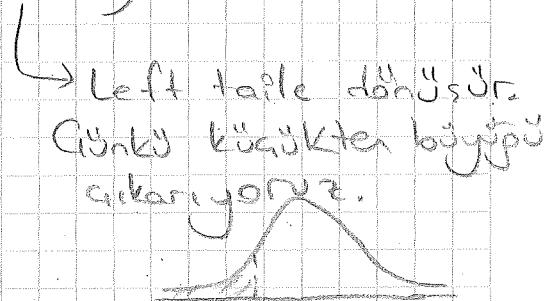
→ one tail olduğu için

output: statistic=3.142, pvalue=0.0068

↙ Gök Mavi çıktı.

! Zher moodpre - moodpost yazsaydim, Kodda
su degisiklikler yapmam gerekiyor:

`stats.ttest_rel(prozac.moodpre, prozac.moodpost,
alternative = "less")`



⑤ Conclusions:

`alpha = 0.05`

`if rel-test < alpha :`

`print ("Reject the null")`

`else :`

`print ("Fail to reject")`

output: Reject the null

"0.05 anlamılık seviyesinde Prozac tedavisinin
depresif bireylerde iyileştirici etki gösterdiği ispat-
lanmıştır."

ttest Formülleri

Independent Sample t Test:

```
stats.ttest_ind(x1, x2, equal_var=True  
                ↓  
                array-like      alternative='two-sided')
```

Dependent Sample t Test:

```
stats.ttest_rel(x1, x2, alternative='two-sided')  
                ↓  
                array-like
```

Levene Test: (Varyanslar eşit mi değil mi? testi)

```
stats.levene(x1, x2)  
                ↓  
                array-like
```