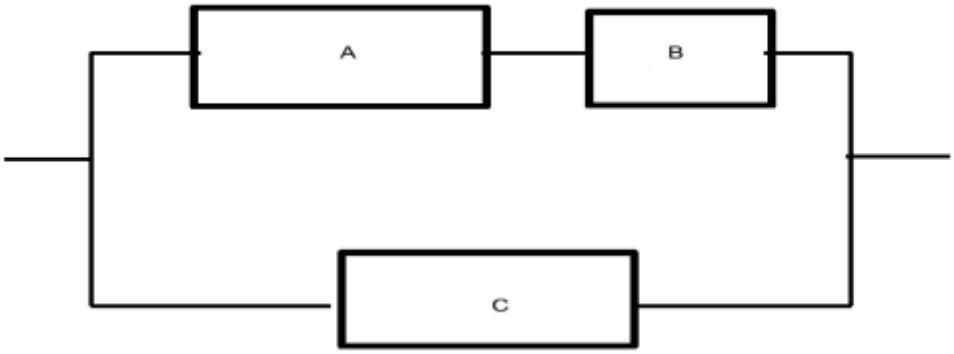
Esercizi

March 27, 2024

1 Esercizi sul calcolo delle probabilità

1.1 Esercizio 1

Un congegno idraulico è costituito da tre valvole collegate come da figura. Le probabilità che le valvole A, B, C si guastino è rispettivamente pari a 1/3, 1/4, 1/2. Il congegno è funzionante se funziona almeno la coppia di valvole A, B o la valvola C. Si calcoli la probabilità che il congegno si guasti.



[21]:
```
GA = 1/3
GB = 1/4
GC = 1/2

G = (GA+GB - (GA*GB))*GC

print("Risultato: ", round(G,2))
```

```
Risultato:  0.25
```

1.2 Esercizio 2

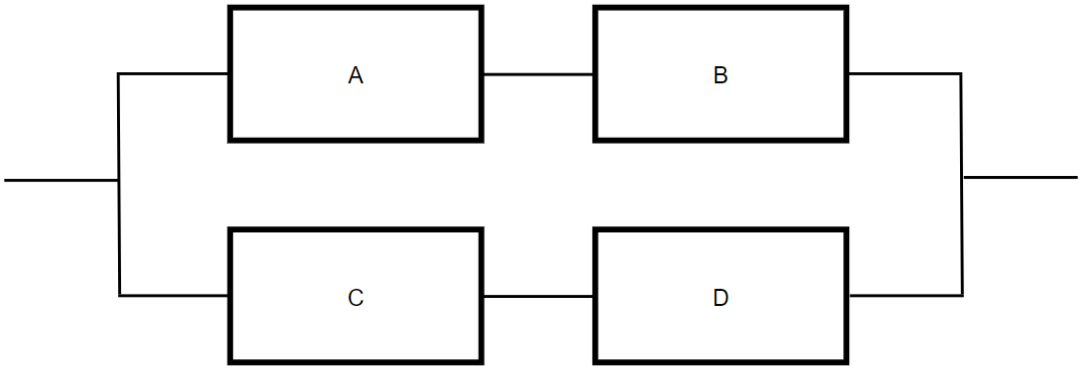
Una industria ha due catene di produzione, A e B, che forniscono rispettivamente il 40% e il 60% dei prodotti. Si scopre che il 25% dei pezzi prodotti dalla catena A sono difettosi contro il 7% dei pezzi prodotti nel centro B. - Qual è la probabilità che un pezzo scelto a caso dall'intera produzione sia difettoso? - Qual è la probabilità che un pezzo difettoso appartenga alla produzione A?

[22]:
```
A = 0.4
B = 0.6

DA = 0.25 # Pezzi prodotti da catena A
DB = 0.07 # Pezzi prodotti da catena B

# Probabilità di trovare un pezzo difettoso (Teorema delle probabilità totali)

PD = (A*DA)+(B*DB)

print("Probabilità che un pezzo scelto a caso dell'intera produzione sia␣
 ↪difettoso: ", round(PD,3))

# Probabilità che un pezzo difettoso appartenga alla catena di produzione A␣
 ↪(Teorema di Bayes)

PAD = (DA*A)/PD

print("Probabilità che un pezzo difettoso appartenga alla catena di produzione␣
 ↪A: ", round(PAD, 3))
```

```
Probabilità che un pezzo scelto a caso dell'intera produzione sia difettoso:
0.142
Probabilità che un pezzo difettoso appartenga alla catena di produzione A:
0.704
```

2 Esercizio 3

In una rete idrica vi sono quattro valvole. Ciascuna valvola ha probabilità 1/4 di guastarsi in un anno. Quando una valvola si guasta impedisce il flusso di acqua. Le valvole funzionano indipendentemente l'una dall'altra. Due valvole sono collegate in serie e queste due sono collegate a loro volta in parallelo con le altre due. Qual è la probabilità che la rete smetta di fornire acqua?



[23]:
```
GA = GB = GC = GD = 1/4

P = (GA+GB-(GA*GB))*(GC+GD-(GC*GD))

print("Risultato: ", round(P,2))
```

```
Risultato:  0.19
```

3 Esercizio 4

Per determinare una password di 3 caratteri si hanno a disposizione le lettere A, B, C, D. Provando a caso con tali lettere: - Qual è la probabilità di indovinare la password in un singolo tentativo? - Qual è tale probabilità se le lettere devono essere distinte? - Qual è tale probabilità se le lettere devono essere distinte e non conta l'ordine?

[24]:
```
import math as m

n = 4
r = 3

# Probabilità di indovinare la password in un singolo tentativo

sol1 = 1/(n**r)

print("Probabilità di indovinare la password in un singolo tentativo: ",␣
 ↪round(sol1,2))

sol2 = 1/((m.factorial(n))/(m.factorial(n-r)))
```

```
print("Probabilità di indovinare la password se le lettere devono essere␣
 ↪distinte: ", round(sol2,2))

sol3 = 1/m.comb(n,r)
# sol3 = 1/(m.factorial(n)/(m.factorial(r)*m.factorial(n-r)))

print("Probabilità di indovinare la password se le lettere devono essere␣
 ↪distinte e non conta l'ordine: ", round(sol3,2))
```

```
Probabilità di indovinare la password in un singolo tentativo:  0.02
Probabilità di indovinare la password se le lettere devono essere distinte:
0.04
Probabilità di indovinare la password se le lettere devono essere distinte e non
conta l'ordine:  0.25
```

4 Esercizio 5

Per determinare una password di 3 caratteri si hanno a disposizione le lettere A, B, C, D, E. Provando a caso con tali lettere: - Qual è la probabilità di indovinare la password in un singolo tentativo? - Qual è tale probabilità se si sa che una delle lettere che figurano nella password è la A? - Qual è tale probabilità se si sa che nella password figurano sia la A che la B?

[25]:
```
import math as m

n = 5
r = 3

# Probabilità di indovinare la password in un singolo tentativo

sol1 = 1/(n**r)

print("Probabilità di indovinare la password in un singolo tentativo: ",␣
 ↪round(sol1,3))
```

```
Probabilità di indovinare la password in un singolo tentativo:  0.008
```

Esercizi

March 27, 2024

1 Esercizi proposti dal docente

Da un'urna contenente 4 palline bianche e 3 nere si eseguono due estrazioni con rimpiazzo (cioè la pallina estratta viene subito rimessa nell'urna). 1) Calcolare la probabilità che le due palline estratte siano del medesimo colore. 2) Calcolare la probabilità che almeno una delle due palline estratte sia nera.

[11]:
```
# Calcolare la prob che le due palline estratte siano dello stesso colore
pb = 4 / 7
pn = 3 / 7
k = 2

punto1 = (pb**2) + (pn**2)
display(punto1)

# Calcolare la probabilità che almeno una delle due palline sia nera
punto2 = 1 - (pb**2)
display(punto2)
```

```
0.510204081632653
```

```
0.6734693877551021
```

I componenti prodotto da una certa ditta possono presentare due tipo di difetti, con percentuali del 3% e del 7% rispettivamente. I due tipo di difettosità si possono produrre in momenti diversi della produzione per cui si può assumere che le presenze dell'uno o dell'altro siano indipendenti tra loro. 1) Qual è la probabilità che un componente presenti entrambi i difetti? 2) Qual è la probabilità che un componente sia difettoso (cioè che presenti almeno uno dei due difetti)? 3) Qual è la probabilità che il componente presenti il difetto 1, sapendo che esso è difettoso? 4) Qual è la probabilità che esso presenti uno solo dei due difetti sapendo che esso è difettoso?

[ ]:
```
p1 = 3 / 100
p2 = 7 / 100

punto1 = p1 * p2
display(punto1)

punto2 = p1 + p2 - punto1
display(punto2)
```

```
punto3 = p1/punto2
display(punto3)
```

[ ]:

DistribuzioneBinomiale

March 1, 2024

1 Distribuzione Binomiale

Descrive un fenomeno che avviene secondo uno schema successo-insuccesso in cui si ripetono n prove indipendenti con due soli esiti.

Esempio: "Lancio di una moneta"

Siano: - **n**: Numero di prove - **k**: Numero di successi - **p**: Probabilità di successo nella singola prova - **q**: Probabilità di insuccesso nella singola provba (1-p) - **X**: Variabile aleatoria che conta i successi - **P(X=k)**: Probabilità da valutare

$$\mathrm{P}(k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$$

Media:

$$E[X] = n \cdot p$$

Varianza:

$$VAR(X) = n \cdot p \cdot (1-p)$$

2 Esercizio

Una ditta produce lampadine, di cui il 5% risulta difettoso, e le vende in confezioni da quattro.

1) Qual è la probabilità che in una confezione ci sia una sola lampadina difettosa?
2) Qual è la probabilità che in una confezione ci siano al più due pezzi difettosi?
3) Se ogni scatola contiene 40 pezzi, quanti pezzi difettosi conterrebbe in media?

[ ]:
```
import scipy.special as sp

# Punto 1

p = 5 / 100
q = 1 - p
```

```
n = 4
k = 1

p_es1 = sp.binom(n, k) * p * q ** (n - k)

print(
    "Probabilità che in una confezione ci sia una lampadina difettosa: ",
    round(p_es1, 4),
)
```

```
Probabilità che in una confezione ci sia una lampadina difettosa:  0.1715
```

[ ]:
```
from scipy.stats import binom

# Punto 1 - Alternativa

p_es1 = binom.pmf(k, n, p)

print(
    "Probabilità che in una confezione ci sia una lampadina difettosa: ",
    round(p_es1, 4),
)
```

```
Probabilità che in una confezione ci sia una lampadina difettosa:  0.1715
```

[ ]:
```
# Punto 2

p0 = sp.binom(4, 0) * pow(q, 4)
p1 = sp.binom(4, 1) * p * pow(q, 3)
p2 = sp.binom(4, 2) * pow(p, 2) * pow(q, 2)

p_es2 = p0 + p1 + p2

print(
    "Probabilità che in una confezione ci siano al più due pezzi difettosi: ",
    round(p_es2, 4),
)
```

```
Probabilità che in una confezione ci siano al più due pezzi difettosi:  0.9995
```

[ ]:
```
# Punto 2 - Alternativa

p_es2 = binom.pmf(0, 4, p) + binom.pmf(1, 4, p) + binom.pmf(2, 4, p)

print(
    "Probabilità che in una confezione ci siano al più due pezzi difettosi: ",
    round(p_es2, 4),
)
```

```
Probabilità che in una confezione ci siano al più due pezzi difettosi:  0.9995
```

[ ]:
```python
# Punto 3

media = 40 * p

print("Media di pezzi difettosi in una scatola di 40 pezzi: ", media)
```

```
Media di pezzi difettosi in una scatola di 40 pezzi:  2.0
```

DistribuzioneGeometrica

March 1, 2024

1 Distribuzione Geometrica

La distribuzione geometrica descrive il numero di insuccessi necessari affinché si verifichi il primo successo in una successione di esperimenti bernoulliani indipendenti e identicamente distribuiti.

La mancanza di memoria è una proprietà caratteristica della distribuzione geometrica. La mancanza di memoria esprime il fatto che una variabile di quei due tipi non "ricorda il passato" ma si comporta come se fosse "nuova".

Probabilità:

$$P(X = k) = p \cdot (1 - p)^{k-1}$$

Media:

$$E[X] = \frac{1}{p}$$

Varianza:

$$VAR(X) = \frac{1 - p}{p^2}$$

2 Esercizio

Lancio di un dado non truccato

1) Qual è la probabilità che esca 6 per la prima volta esattamente al terzo lancio?
2) Sapendo che nei primi 3 lanci non si è avuto alcun 6, qual è la probabilità che esca 6 per la prima volta al quinto tentativo?

[ ]:
```
# Punto 1

p = 1 / 6
k = 3

p_es1 = p * (1 - p) ** (k - 1)

print(
    "Probabilità che esca 6 al terzo lancio: ",
```

```
    round(p_es1, 3),
)
```

```
Probabilità che esca 6 al terzo lancio:  0.116
```

[ ]:
```
# Punto 2 (Proprietà di mancanza di memoria)

k = 2

p_es2 = p * (1 - p) ** (k - 1)

print(
    "Probabilità che esca 6 al quinto tentativo: ",
    round(p_es2, 3),
)
```

```
Probabilità che esca 6 al quinto tentativo:  0.139
```

DistribuzioneIpergeometrica

March 1, 2024

1 Distribuzione Ipergeometrica

Viene utilizzata in situazioni del tipo estrazioni da un'urna senza rimpiazzo. L'estrazione è influenzata da quelle precedenti.

Siano: - **b**: Palline bianche uguali - **r**: Palline rosse uguali - **n**: Numero di palline estratte senza rimpiazzo (n<=b+r) - **X**: Variabile aleatoria che conta i successi - **P(X=k)**: Probabilità da valutare

In quanti modi si possono scegliere k palline bianche dalle b bianche?

$$\binom{b}{k}$$

In quanti modi si possono scegliere n-k palline bianche dalle r rosse?

$$\binom{r}{n-k}$$

In quanti modi si possono scegliere n palline bianche dalle b+r totali?

$$\binom{b+r}{n}$$

Probabilità:

$$\mathrm{P}(k) = \frac{\binom{b}{k} \cdot \binom{r}{n-k}}{\binom{b+r}{n}}$$

Media:

$$E[X] = n \cdot \frac{b}{b+r}$$

Varianza:

$$VAR(X) = n \cdot \frac{br}{(b+r)^2} \cdot \frac{b+r-n}{b+r-1}$$

2 Esercizio

La memoria di un computer è composta da 30 hard disk, ognuno dei quali contiene 100 file. Un programma dovrà accedere a 28 di questi file (tutti diversi)

1) Qual è la probabilità che non ci siano file provenienti dall'hard disk n°1?
2) Qual è la probabilità del punto 1 nel caso in cui i file possano ripetersi?

[ ]:
```
from scipy.special import binom

# Punto 1

b = 100
r = 2900
n = 28

p_es1 = binom(b, 0) * binom(r, n) / binom(b + r, n)

print(
    "Probabilità che non ci siano file provenienti dall'hard disk 1: ",
    round(p_es1, 3),
)
```

```
Probabilità che non ci siano file provenienti dall'hard disk 1:  0.385
```

[ ]:
```
# Punto 2 (X segue una legge binomiale)

p_es2 = binom(n, 0) * pow(1 / 30, 0) * pow(1 - 1 / 30, 28)

print(
    "Probabilità nel caso in cui i file possano ripetersi: ",
    round(p_es2, 3),
)
```

```
Probabilità nel caso in cui i file possano ripetersi:  0.387
```

DistribuzioneMultinomiale

March 1, 2024

1 Distribuzione Multinomiale

La distribuzione multinomiale è una generalizzazione della distribuzione binomiale, utilizzata quando ci sono più di due possibili esiti (o categorie) per ogni singola prova. In una distribuzione multinomiale, ciascuna prova può cadere in una delle k categorie con probabilità p1, p2, …, pk, dove p1 + p2 + … + pk = 1 .

La probabilità di ottenere un vettore di conteggi x1, x2, …, xk, dove xi è il numero di volte che l'evento della categoria i si verifica, è data da:

dove: - **n**: è il numero totale di prove. - **xi**: è il numero di volte che l'evento della categoria i si verifica. - **pi**: è la probabilità di ottenere l'evento della categoria i.

Probabilità:

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k) = \frac{n!}{x_1! \cdot x_2! \cdot \ldots \cdot x_k!} \cdot p_1^{x_1} \cdot p_2^{x_2} \cdot \ldots \cdot p_k^{x_k}$$

2 Esercizio

Calcolare la probabilità che lanciando un dado non truccato quattro volte, esca tre volte 6 ed una volta 2

[ ]:
```
# Il fenomeno segue la legge multinomiale B(4, 1/6, 1/6, 1/6, 1/6, 1/6, 1/6)
# P(Y=(0, 1, 0, 0, 0, 3))

from scipy.special import binom
from math import factorial

p = 1 / 6

p_es1 = (factorial(4) / factorial(3)) * (p**4)

print(
    "Probabilità che lanciando un dado non truccato quattro volte, esca tre␣
 ↪volte 6 ed una volta 2: ",
    round(p_es1, 3),
)
```

```
Probabilità che lanciando un dado non truccato quattro volte, esca tre volte 6
ed una volta 2:  0.003
```

DistribuzionePoisson

March 1, 2024

1 Distribuzione Poisson

La distribuzione di Poisson si utilizza quando in uno schema successo-insuccesso si ha un alto numero di prove con una probabilità base di successo in ogni singola prova.

Probabilità:

$$P(X = k) = e^{-\lambda} \cdot \frac{\lambda^k}{k!}$$

Media:

$$E[X] = \lambda$$

Varianza:

$$VAR(X) = \lambda$$

2 Esercizio

In un libro di 500 pagine sono distribuiti a caso 300 errori di stampa

Qual è la probabilità che una data pagina contenga almeno due errori?

[ ]:
```
from math import e
from math import factorial

n = 300
p = 1 / 500
lmd = n * p  #lambda

p0 = e ** (-lmd) * ((lmd**0) / (factorial(0)))
p1 = e ** (-lmd) * ((lmd**1) / (factorial(1)))

p_es1 = 1 - p0 - p1

print(
    "Probabilità che una data pagina contenga almeno due errori: ",
    round(p_es1, 3),
```

```
)
```

```
Probabilità che una data pagina contenga almeno due errori:  0.122
```

esercizi

March 27, 2024

1 Esercizi sulle variabili aleatorie discrete

1.1 Esercizio 1

Due centralini, tra di loro indipendenti, ricevono nell'unità di tempo un numero di telefonate X e Y aventi legge di Poisson di parametri rispettivamente e . 1) Qual è la probabilità che nell'unità di tempo i due centralini ricevano insieme non più di tre telefonate, supponendo =2 e =4? 2) Calcolare la legge condizionale di X dato X+Y=n. Si tratta di una densità nota? Quanto vale la media di questa legge condizionale? 3) Supponendo =2 e =4 e sapendo che nell'unità di tempo i due centralini hanno ricevuto complessivamente 8 telefonate, qual è la probabilità che il primo ne abbia ricevute 3?

[290]:
```
from scipy.stats import poisson

lam = 2
mu = 4
punto1 = (
    poisson.pmf(0, lam + mu)
    + poisson.pmf(1, lam + mu)
    + poisson.pmf(2, lam + mu)
    + poisson.pmf(3, lam + mu)
)

punto1
```

[290]:
```
0.15120388277664787
```

[291]:
```
from scipy.stats import binom

n = 8
k = 3
punto3 = binom.pmf(k, n, lam / (lam + mu))
punto3
```

[291]:
```
0.27312909617436376
```

1.2 Esercizio 2

Da una rilevazione risulta che il numero di incidenti stradali che avvengono ad un determinato incrocio in un mese segue una distribuzione di Poisson con valor medio 1.5. 1) Qual è la probabilità che in un mese non ci siano incidenti? 2) Qual è la probabilità che in un mese ci siano più di due incidenti?

[292]:
```
from scipy.stats import poisson

lam = 1.5
punto1 = poisson.pmf(0, lam)
punto1
```

[292]:
```
0.22313016014842982
```

[293]:
```
punto2 = 0
for i in range(3):
    punto2 += poisson.pmf(i, lam)

punto2 = 1 - punto2
punto2
```

[293]:
```
0.19115316946194194
```

1.3 Esercizio 3

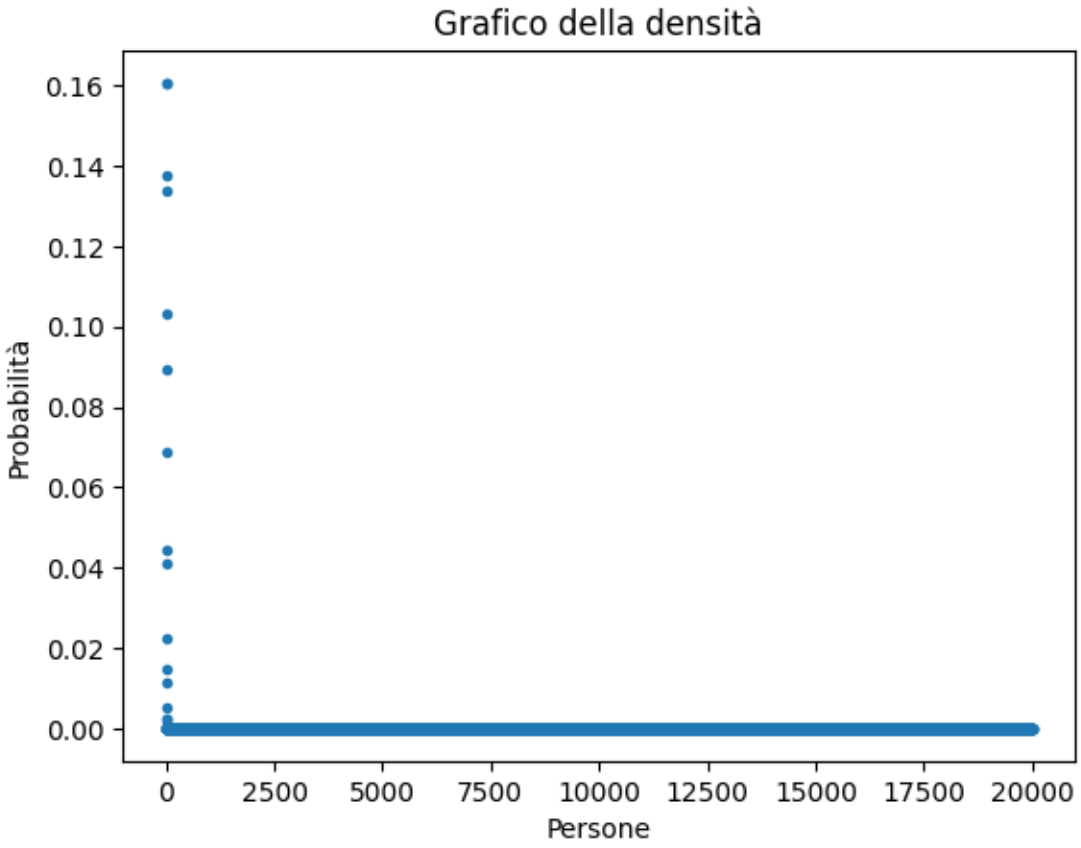
La probabilità di contrarre una malattia rara è dello 0.03%. Qual è la probabilità che in una città dove vivono 20000 persone vi siano meno di 4 persone che contraggono la malattia? Costruire il grafico della densità e della funzione di ripartizione della distribuzione in esame.

[294]:
```
from scipy.stats import poisson

p = 3 / 10000
n = 20000
lam = p * n
punt1 = 0
for i in range(4):
    punt1 += poisson.pmf(i, lam)

punt1
```

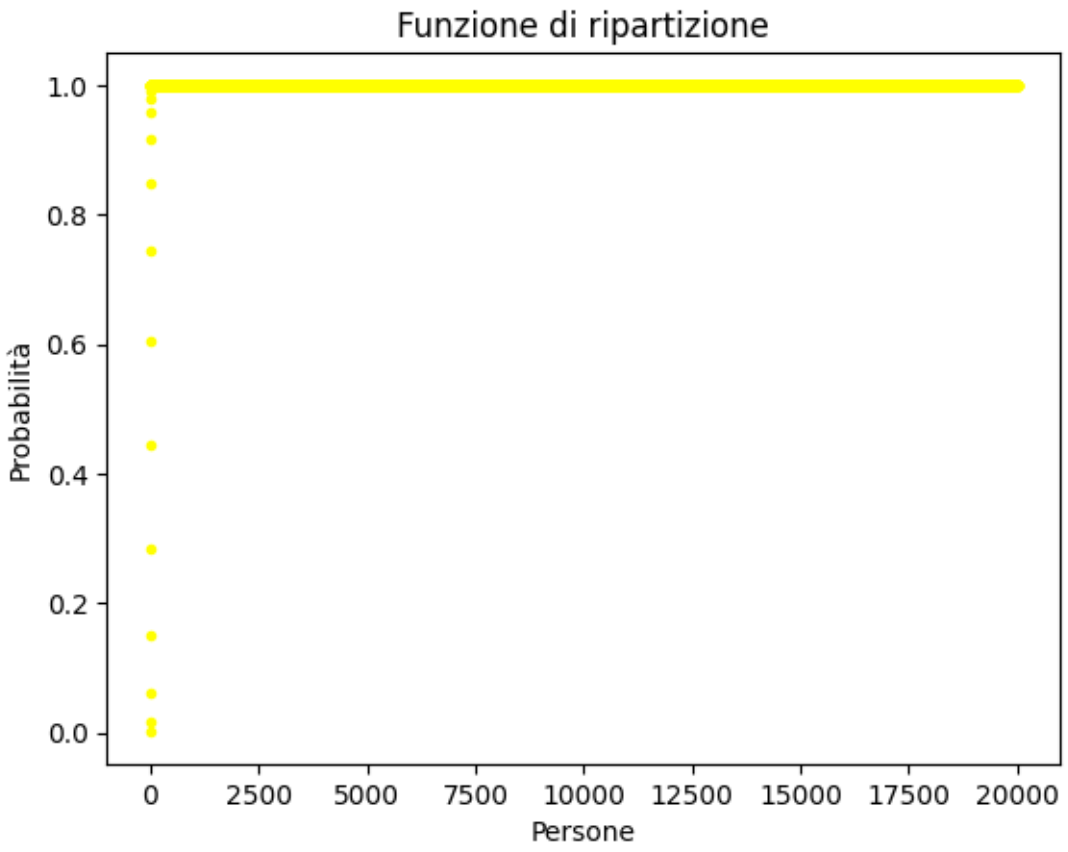
[294]:
```
0.15120388277664804
```

[295]:
```
import matplotlib.pyplot as plt

x = range(0, n + 1)
y = poisson.pmf(x, lam)
plt.plot(x, y, ".")
plt.xlabel("Persone")
```

```
plt.title("Grafico della densità")
plt.ylabel("Probabilità")
plt.show()
```



[296]:
```
import matplotlib.pyplot as plt

x = range(0, n + 1)
y = poisson.cdf(x, lam)
plt.plot(x, y, ".", color="yellow")
plt.xlabel("Persone")
plt.title("Funzione di ripartizione")
plt.ylabel("Probabilità")
plt.show()
```



1.4 Esercizio 4

Si lancia un dado equilibrato finché non esca un numero dispari. 1) Qual è la probabilità che ciò avvenga al quarto tentativo? 2) Quanti tentativi sono necessari affinché si abbia una probabilità maggiore del 95% che esca un numero dispari esattamente al tentativo successivo? 3) Sapendo che nei primi 4 lanci non si è avuto un numero dispari, qual è la probabilità che si abbia un numero dispari per la prima volta al settimo tentativo?

[297]:
```
from scipy.stats import geom

x = 4
p = 1 / 2

puntoa = geom.pmf(x, p)
puntoa
```

[297]: 0.0625

[298]:
```
from scipy.stats import geom
```

```
k = 1
while geom.cdf(k, p) < (95 / 100):
    k += 1
k
```

[298]: 5

[299]:
```
from scipy.stats import geom

x = 3
p = 1 / 2

puntoc = geom.pmf(x, p)
puntoc
```

[299]: 0.125

1.5 Esercizio 5

Si supponga che tre negozi della stessa tipologia attraggano rispettivamente il 20% della clientela, il 45% e il 35%. 1) Scegliendo a caso 6 clienti, qual è la probabilità che 2 vadano nel primo negozio, 1 nel secondo e 3 nel terzo? 2) Qual è la probabilità che nessun cliente vada nel primo negozio?

[300]:
```
from scipy.stats import multinomial

p = [0.20, 0.45, 0.35]
n = [2, 1, 3]
punto1 = multinomial.pmf(n, sum(n), p)
punto1
```

[300]: 0.04630500000000001

[301]:
```
from scipy.stats import binom

punto2 = binom.pmf(0, 6, 0.2)
punto2
```

[301]: 0.2621440000000001

DistribuzioneNormale

March 27, 2024

1 Distribuzione Normale

La distribuzione normale, o gaussiana, è fondamentale in statistica e probabilità per modellare una vasta gamma di fenomeni naturali e artificiali. È utilizzata in analisi statistica, processi di controllo di qualità, finanza, ingegneria e altri campi, poiché approssima molte distribuzioni di dati reali ed è descritta da proprietà ben definite come media e deviazione standard.

In tal caso scriviamo:

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Dove:

- x: la variabile aleatoria,
- μ: la media della distribuzione,
- σ: la deviazione standard.

Funzione di densità di probabilità (PDF)

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Media:

$$E[X] = \mu$$

Varianza:

$$VAR(X) = \sigma^2$$

1.1 Distribuzione Normale Standard

Si denota con:

$$X \sim \mathcal{N}(0, 1)$$

Funzione di densità di probabilità (PDF)

$$f(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}}$$

2 Esercizio

Un lago riceve acqua da due immissari ed alimenta un emissario. Misurando la portata in base alla variazione dell'acqua, i due immissari immettono con legge X1 N(1,1) ed X2 N(2,2), mentre l'emissario viene alimentato con legge X3 N(3/2,3).

1) Si determini la legge seguita dall'altezza dell'acqua
2) Qual è la probabilità che la quota superi il livello di guardia pari a 2?
3) Qual è la probabilità che la quota sia inferiore a 0.5

[15]:
```
from scipy.stats import norm
from math import sqrt

# Punto 1

# Medie
e1 = 1
e2 = 2
e3 = 3/2

# Varianze
v1 = 1
v2 = 2
v3 = 3

media = e1 + e2 - e3
varianza = v1 + v2 + v3

print("La legge seguita dall'altezza dell'acqua è X ~ N({}, {})".format(media,␣
 ↪varianza))
```

```
La legge seguita dall'altezza dell'acqua è X ~ N(1.5, 6)
```

[16]:
```
"""
    x: rappresenta il valore o i valori per i quali si desidera calcolare la␣
 ↪funzione di densità di probabilità
    loc: media
    scale: deviazione standard

    norm.pdf(x,loc,scale): Ritorna la probabilità che una variabile casuale X␣
 ↪assuma un valore specifico x. La PDF è utilizzata quando siamo interessati␣
 ↪al risultato esatto.
    norm.cdf(x,loc,scale): Ritorna la probabilità che una variabile casuale X␣
 ↪sia minore o uguale a un valore specifico x. La CDF è utilizzata quando␣
 ↪siamo interessati alla probabilità cumulativa fino a un certo punto.
```

```
"""

# Punto 2

punto2 = 1 - norm.cdf(2, media, sqrt(varianza))

print(
    "Probabilità che la quota superi il livello di guardia pari a 2: ",
    round(punto2, 2),
)
```

```
Probabilità che la quota superi il livello di guardia pari a 2:  0.42
```

[17]:
```
# Punto 3

punto3 = norm.cdf(0.5, media, sqrt(varianza))

print(
    "Probabilità che la quota sia inferiore a 0.5: ",
    round(punto3, 2),
)
```

```
Probabilità che la quota sia inferiore a 0.5:  0.34
```

esercizi

March 27, 2024

1 Esercizi sulle variabili aleatorie continue

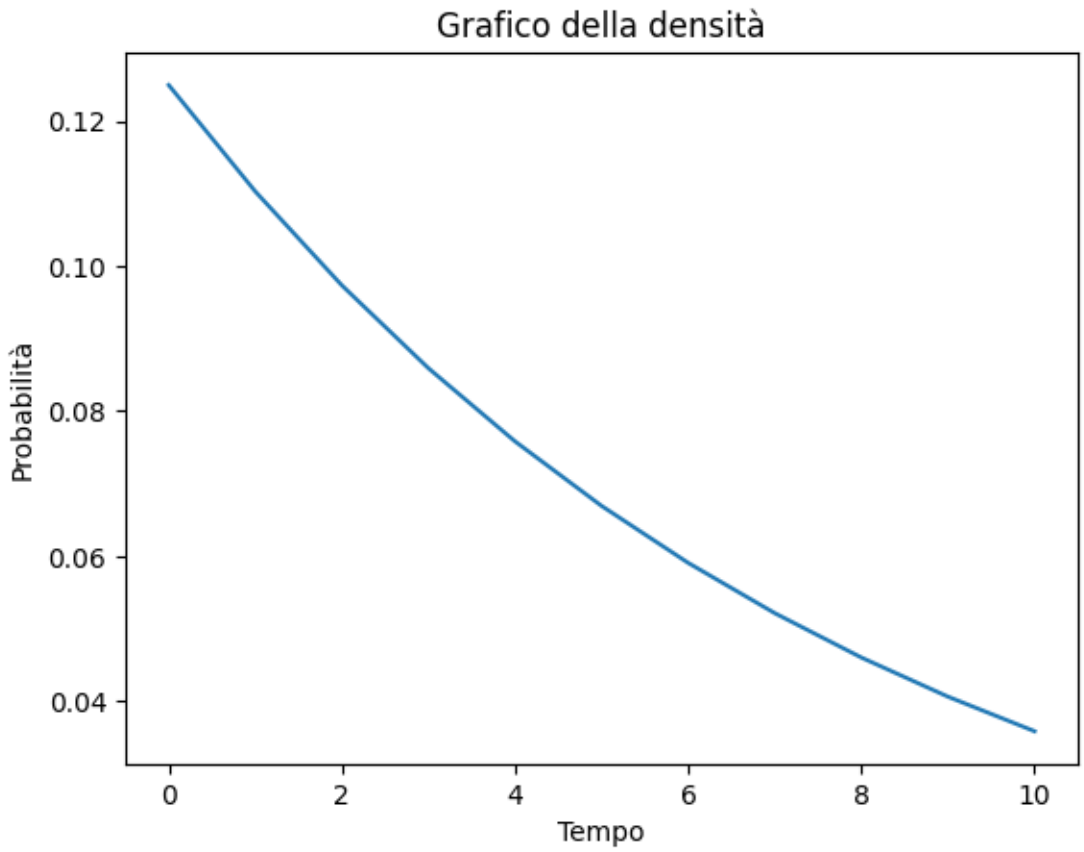
1.1 Esercizio 1

Il numero di anni di funzionamento di una radio ha una distribuzione esponenziale di parametro =1/8. Qual è la probabilità che una radio funzioni per più di dieci anni? Costruire il grafico della densità e della funzione di ripartizione della distribuzione in esame

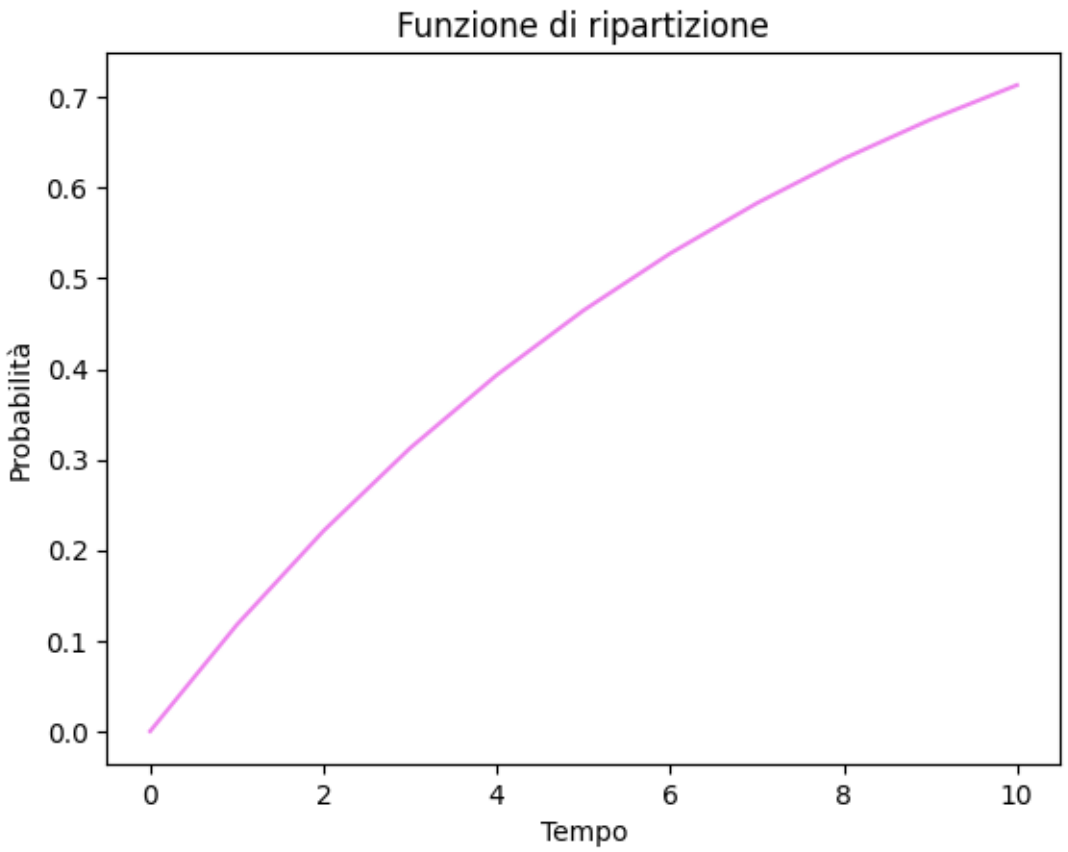
[338]:
```
from scipy.stats import expon

lam = 1 / 8
x = 10
punto1 = 1 - expon.cdf(x, 0, 1 / lam)
punto1
```

[338]:
```
0.28650479686019015
```

[339]:
```
import matplotlib.pyplot as plt

assex = range(11)
assey = expon.pdf(assex, 0, 1 / lam)
plt.plot(assex, assey)
plt.xlabel("Tempo")
plt.ylabel("Probabilità")
plt.title("Grafico della densità")
plt.show()
```



[340]:
```
import matplotlib.pyplot as plt

assex = range(11)
assey = expon.cdf(assex, 0, 1 / lam)
plt.plot(assex, assey, color="violet")
plt.xlabel("Tempo")
plt.ylabel("Probabilità")
plt.title("Funzione di ripartizione")
plt.show()
```



1.2 Esercizio 2

Il tempo (in ore) necessario per riparare un macchinario è una v.a. esponenziale di parametro =1. 1) Qual è la probabilità che la riparazione superi le due ore di tempo? 2) Qual è la probabilità che la riparazione richieda almeno tre ore, sapendo che ne richiede più di due?

[341]:
```
from scipy.stats import expon

lam = 1
x = 2
punto1 = 1 - expon.cdf(x, 0, 1 / lam)
punto1
```

[341]:
```
0.1353352832366127
```

[342]:
```
from scipy.stats import expon

lam = 1
x = 1
punto2 = 1 - expon.cdf(x, 0, 1 / lam)
```

```
punto2
```

[342]: 0.36787944117144233

1.3 Esercizio 3

Si suppone che l'altezza degli uomini in Italia segua approssimativamente una v.a. normale di media 175 cm e deviazione standard 9 cm.

1) Quale sarebbe la percentuale di italiani di statura superiore al metro e 90?

Alla visita di leva vengono scartate le reclute di altezza inferiore ai 153 cm.

2) Quale sarebbe la percentuale di reclute scartate alla visita di leva?
3) Costruire il grafico della densità e della funzione di ripartizione della distribuzione in esame.

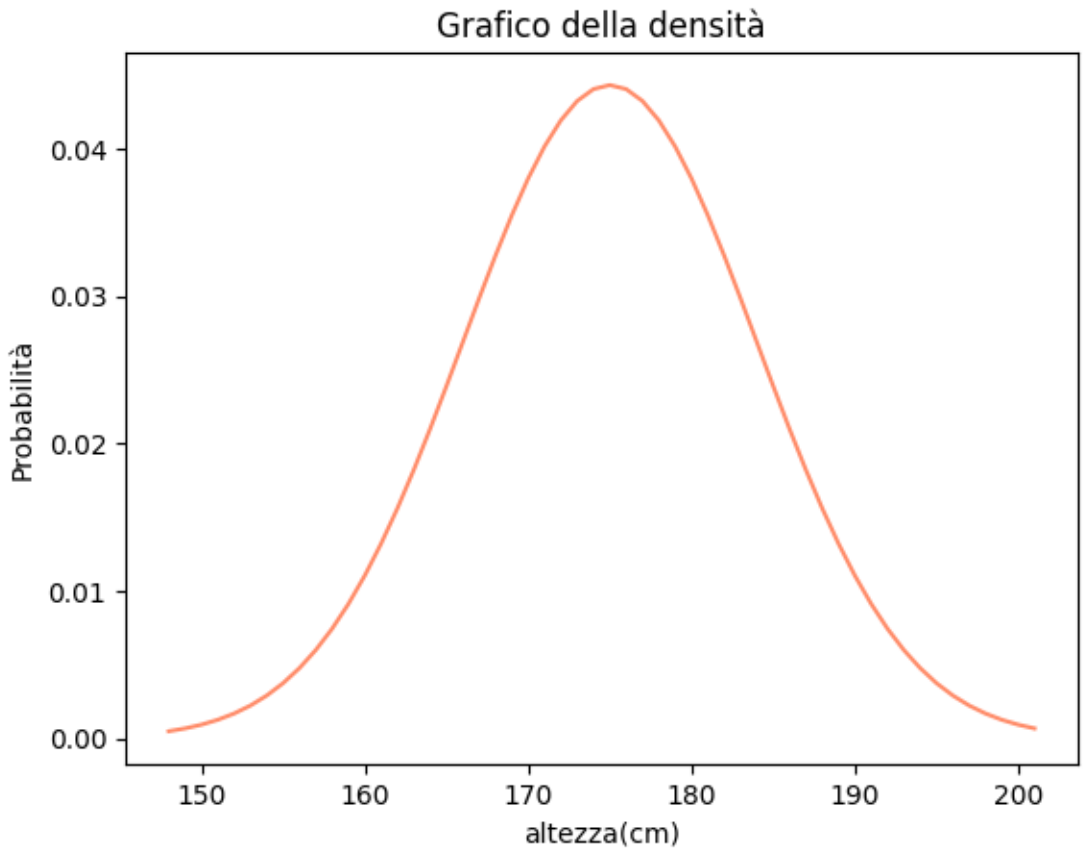
[343]:
```
from scipy.stats import norm

med = 175
dev = 9

punto1 = 1 - (norm.cdf(190, med, dev))
punto1
```

[343]: 0.047790352272814696

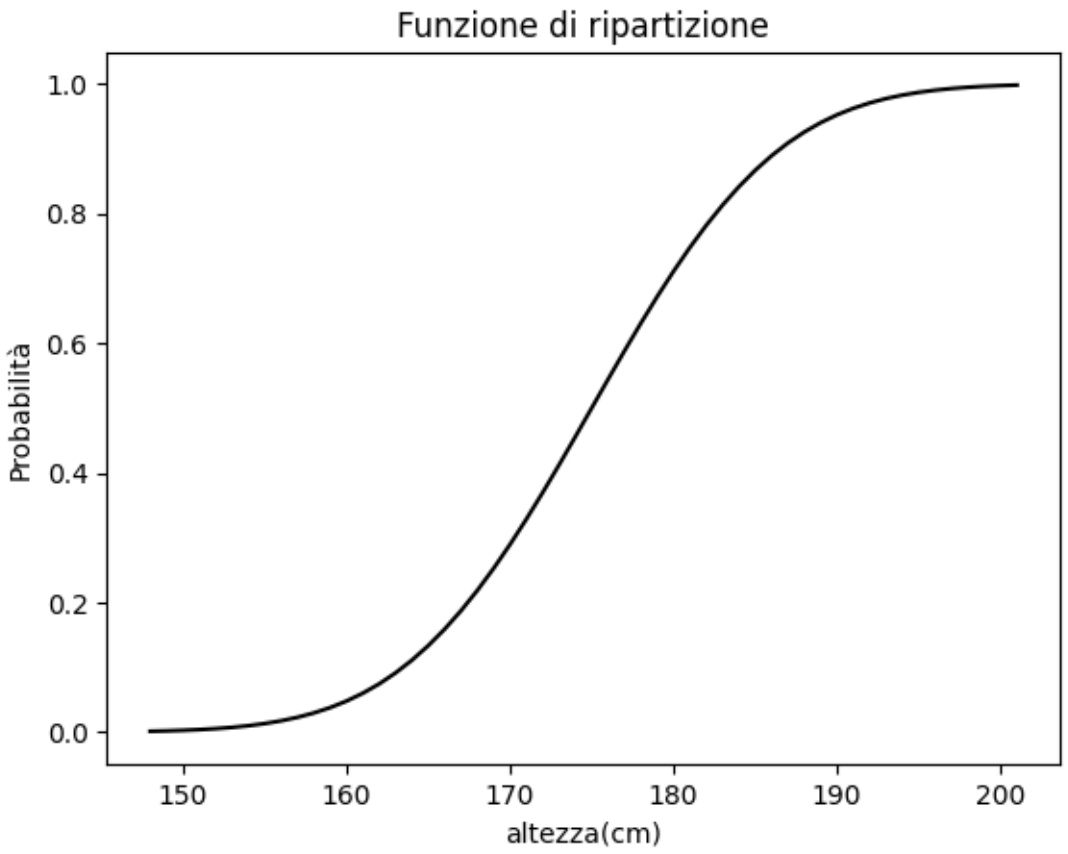
[344]:
```
from scipy.stats import norm

med = 175
dev = 9

punto2 = norm.cdf(153, med, dev)
punto2
```

[344]: 0.007253771124867817

[345]:
```
import matplotlib.pyplot as plt
import numpy as np

assex = np.arange(med - (3 * dev), med + (3 * dev))
assey = norm.pdf(assex, med, dev)
plt.plot(assex, assey, color="#ff8c69")
plt.xlabel("altezza(cm)")
plt.ylabel("Probabilità")
plt.title("Grafico della densità")
plt.show()
```



[346]:
```
import matplotlib.pyplot as plt
import numpy as np

assex = np.arange(med - (3 * dev), med + (3 * dev))
assey = norm.cdf(assex, med, dev)
plt.plot(assex, assey, color="black")
plt.xlabel("altezza(cm)")
plt.ylabel("Probabilità")
plt.title("Funzione di ripartizione")
plt.show()
```



1.4 Esercizio 4

Si scelga a caso un punto X all'interno dell'intervallo [0,2].

Qual è la probabilità che il triangolo equilatero il cui lato ha lunghezza X abbia area maggiore di 1?

[347]:
```
from scipy.stats import uniform

a = 0
b = 2
x = np.sqrt(4 / (np.sqrt(3)))
puntounico = 1 - (uniform.cdf(x, a, b - a))
puntounico
```

[347]: 0.24016431434840735

1.5 Esercizio 5

Ad un esame universitario, il voto medio è stato 24 e la deviazione standard 4. Supponendo i voti distribuiti normalmente, calcolare: 1) la probabilità che uno studente abbia riportato un

voto superiore a 27 2) la probabilità che uno studente abbia riportato un voto inferiore a 22 3) la probabilità che uno studente abbia riportato un voto compreso tra 23 e 25 4) il voto minimo riportato dal 70% degli studenti 5) il voto massimo non superato dal 90% degli studenti

[348]:
```
from scipy.stats import norm

med = 24
dev = 4
punto1 = 1 - (norm.cdf(27, med, dev))
punto1
```

[348]:
```
0.22662735237686826
```

[349]:
```
from scipy.stats import norm

med = 24
dev = 4
punto2 = norm.cdf(22, med, dev)
punto2
```

[349]:
```
0.3085375387259869
```

[350]:
```
from scipy.stats import norm

med = 24
dev = 4
p23 = norm.cdf(23, med, dev)
p25 = norm.cdf(25, med, dev)
punto3 = p25 - p23
punto3
```

[350]:
```
0.1974126513658474
```

[351]:
```
from scipy.stats import norm

punto4 = norm.ppf(1 - 0.7, med, dev)
punto4
```

[351]:
```
21.902397949167838
```

[352]:
```
from scipy.stats import norm

punto5 = norm.ppf(0.9, med, dev)
punto5
```

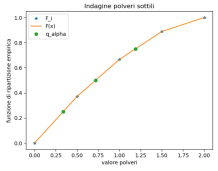
[352]:
```
29.1262062621784
```

Esercizio 1

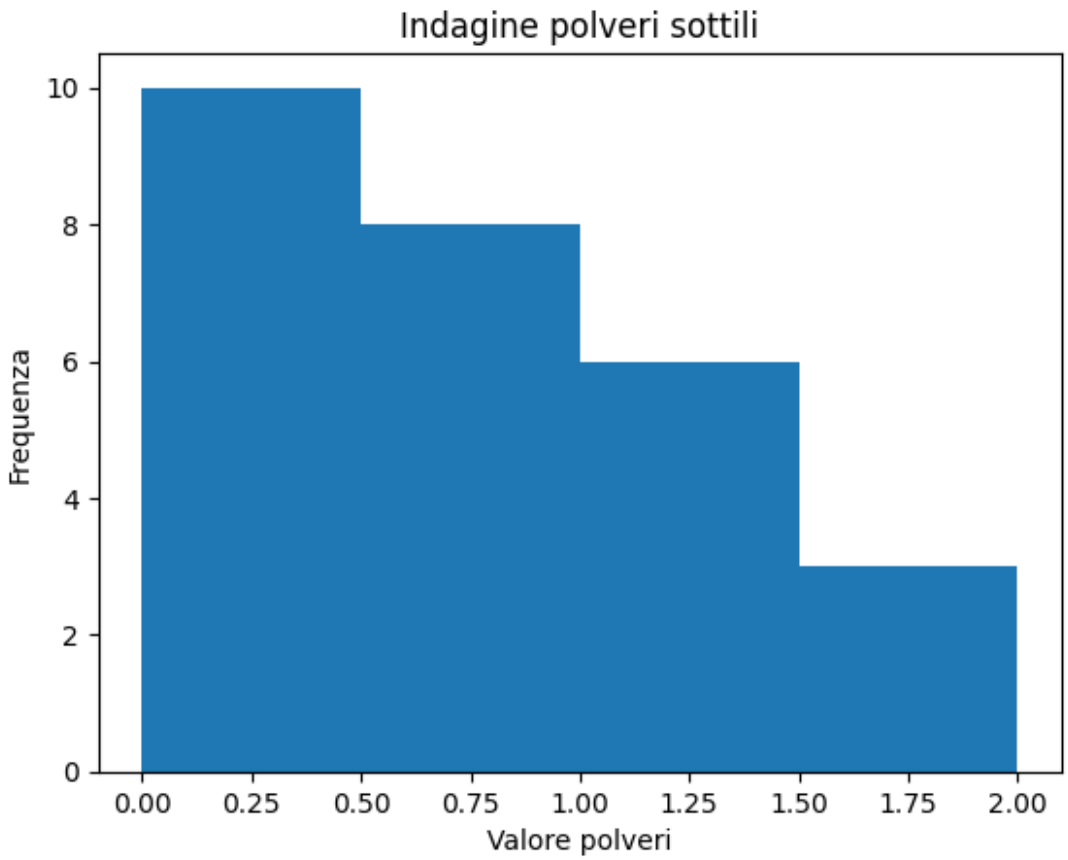
Un monitoraggio sulla densità di polveri sottili nell'aria ha condotto alle frequenze riportate nella tabella sotto, in opportune unità di misura. Si calcolino i quartili empirici.

classi	frequenze
$[0, 0.5[$	10
$[0.5, 1[$	8
$[1, 1.5[$	6
$[1.5, 2]$	3

Diagramma




```
import matplotlib.pyplot as plt
import numpy as np

assex = np.array([[0, 0.5], [0.5, 1], [1, 1.5], [1.5, 2]])
assey = [10, 8, 6, 3]

ampiezza = assex[:, 1] - assex[:, 0]
centers = 0.5 * (assex[:, 1] + assex[:, 0])

probabilita = assey / np.sum(assey)  # pi
sommacumulativa = np.cumsum(probabilita)  # Fi
sommacumulativa = np.insert(sommacumulativa, 0, 0)  # Inserisce 0
all'inizio dell'array

plt.bar(centers, assey, ampiezza)

plt.title("Indagine polveri sottili")
plt.xlabel("Valore polveri")
plt.ylabel("Frequenza")

plt.show()
```




```
x = [0, 0.5, 1, 1.5, 2]
alpha = [0.25, 0.5, 0.75]

# Calcolo degli indici per la funzione di ripartizione empirica

indici = np.zeros(3, dtype=int)

for k in range(3):
    for j in range(1, 5):
        if alpha[k] > sommacumulativa[j - 1] and alpha[k] <=
sommacumulativa[j]:
            indici[k] = j

print(indici)
```

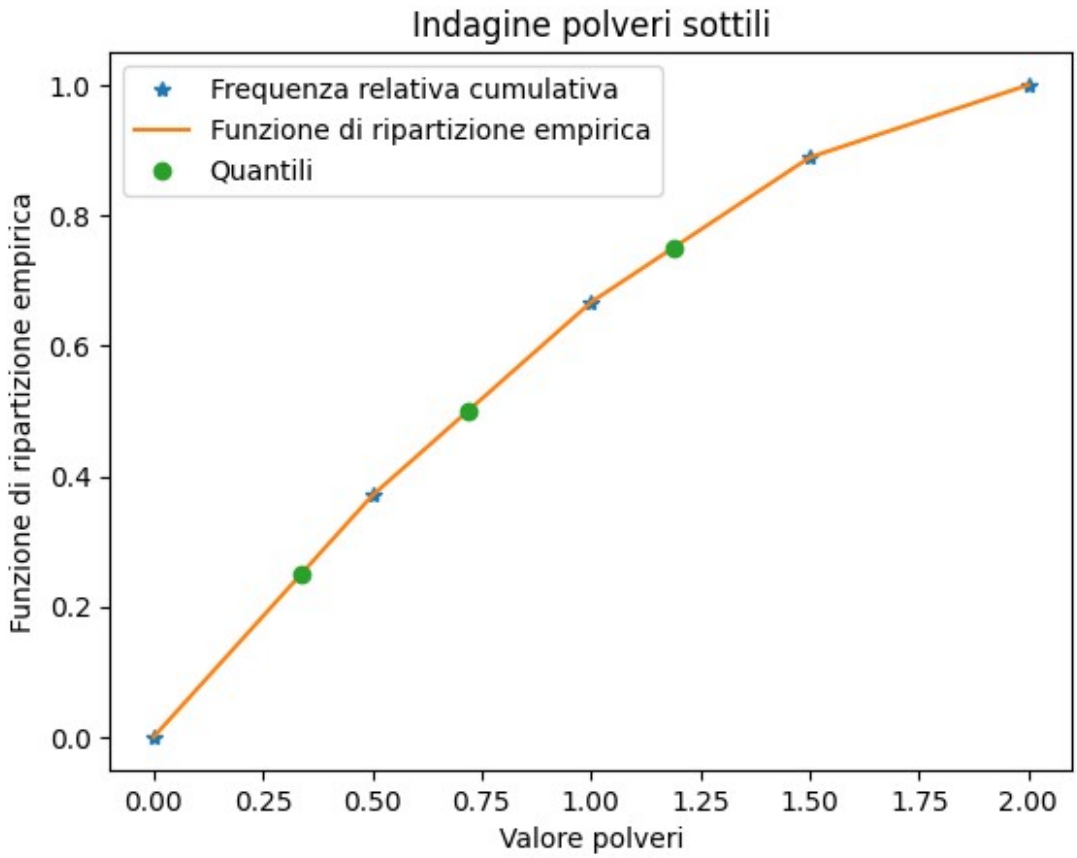
```
[1 2 3]
```

```
# Calcolo dei quantili

q_alpha = np.zeros(3)

for k in range(3):
    j = indici[k]
    q_alpha[k] = x[j - 1] + (alpha[k] - sommacumulativa[j - 1]) *
```

```
(x[j] - x[j - 1]) / (
        sommacumulativa[j] - sommacumulativa[j - 1]
    )

print(q_alpha)
```

```
[0.3375  0.71875 1.1875 ]
```

```
# Creazione del grafico

plt.plot(x, sommacumulativa, "*", label="Frequenza relativa
cumulativa")
plt.plot(x, sommacumulativa, label="Funzione di ripartizione
empirica")
plt.plot(q_alpha, alpha, "o", label="Quantili")

plt.legend()

plt.title("Indagine polveri sottili")
plt.xlabel("Valore polveri")
plt.ylabel("Funzione di ripartizione empirica")

plt.show()
```

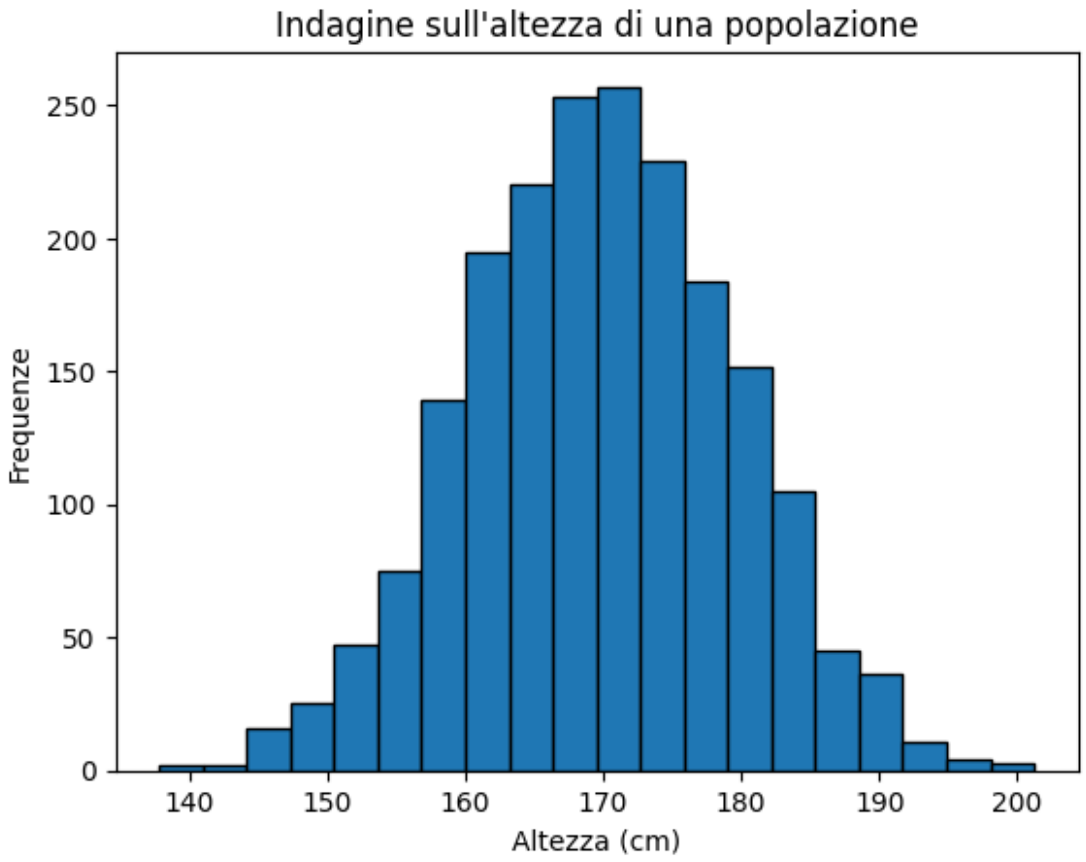


Esercizio 2

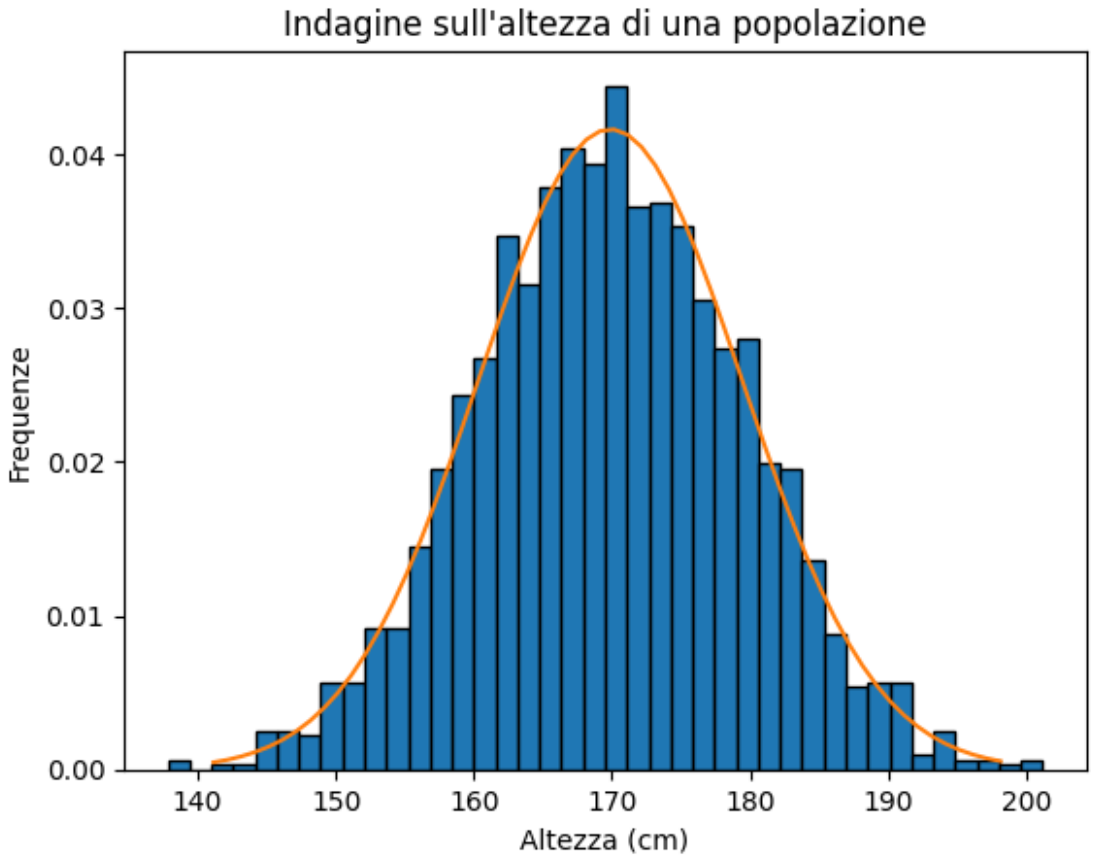
L'altezza di 2000 individui di una popolazione è riportata nel file 'Data altezze.dat'.

1. Calcolare media e deviazione standard della popolazione.
2. Costruire un istogramma a 20 barre.
3. Si possono adattare ai dati una distribuzione normale?

```
import numpy as np
import matplotlib.pyplot as plt

X = np.loadtxt("Data_altezze.dat")

media = np.mean(X)
dev_std = np.std(X)

plt.hist(X, 20, edgecolor="black")

plt.title("Indagine sull'altezza di una popolazione")
plt.xlabel("Altezza (cm)")
plt.ylabel("Frequenze")

plt.show()
```



```
from scipy.stats import norm

assex = np.arange(media - (3 * dev_std), media + (3 * dev_std))
assey = norm.pdf(assex, media, dev_std)

plt.hist(X, 40, edgecolor="black", density="True")
plt.plot(assex, assey)

plt.title("Indagine sull'altezza di una popolazione")
plt.xlabel("Altezza (cm)")
plt.ylabel("Frequenze")

plt.show()
```



Esercizio 3

Si vuole testare un dispositivo con uno strumento che fornice delle misure di voltaggio. Si eseguono 9 misurazioni registrando i valori in volt 11, 13.2, 12.3, 10.9, 13, 10.5, 12.3, 13, 13.15. E' nota la precisione dello strumento e si ha $\sigma = 1$ V.

1. Si determinino gli intervalli di confidenza al 95% e al 99%.
2. Determinare gli stessi intervalli di confidenza nel caso in cui si avesse $\sigma = 1.4$ V.

$$11,\ 13.2,\ 12.3,\ 10.9,\ 13,\ 10.5,\ 12.3,\ 13,\ 13.15.$$

3. Sempre con precisione σ = 1 V, determinare gli stessi intervalli con la stessa media delle misure ma supponendo che essa provenga da un campione di 20 misurazioni.

```
import numpy as np
from scipy.stats import norm

X = np.array([11, 13.2, 12.3, 10.9, 13, 10.5, 12.3, 13, 13.15])
media = np.mean(X)
n = X.size

# Primo punto

sigma = 1

alpha = 0.05
phi = norm.ppf(1 - alpha / 2)

min = media - sigma / np.sqrt(n) * phi
max = media + sigma / np.sqrt(n) * phi

print("Intervallo di confidenza al 95% [{},{}]".format(min, max))

alpha = 0.01
phi = norm.ppf(1 - alpha / 2)

min = media - sigma / np.sqrt(n) * phi
max = media + sigma / np.sqrt(n) * phi

print("Intervallo di confidenza al 99% [{},{}]".format(min, max))
```

```
Intervallo di confidenza al 95%
[11.496678671819982,12.803321328180019]
Intervallo di confidenza al 99%
[11.291390232150366,13.008609767849634]
```

```
# Secondo punto

sigma = 1.4

alpha = 0.05
phi = norm.ppf(1 - alpha / 2)

min = media - sigma / np.sqrt(n) * phi
max = media + sigma / np.sqrt(n) * phi

print("Intervallo di confidenza al 95% [{},{}]".format(min, max))

alpha = 0.01
phi = norm.ppf(1 - alpha / 2)

min = media - sigma / np.sqrt(n) * phi
max = media + sigma / np.sqrt(n) * phi
```

```
print("Intervallo di confidenza al 99% [{},{}]".format(min, max))
```

```
Intervallo di confidenza al 95%
[11.235350140547975,13.064649859452025]
Intervallo di confidenza al 99%
[10.947946325010513,13.352053674989488]
```

```
# Terzo punto

sigma = 1
n = 20

alpha = 0.05
phi = norm.ppf(1 - alpha / 2)

min = media - sigma / np.sqrt(n) * phi
max = media + sigma / np.sqrt(n) * phi

print("Intervallo di confidenza al 95% [{},{}]".format(min, max))

alpha = 0.01
phi = norm.ppf(1 - alpha / 2)

min = media - sigma / np.sqrt(n) * phi
max = media + sigma / np.sqrt(n) * phi

print("Intervallo di confidenza al 99% [{},{}]".format(min, max))
```

```
Intervallo di confidenza al 95% [11.71173872971171,12.58826127028829]
Intervallo di confidenza al 99%
[11.574027057882873,12.725972942117128]
```

Esercizio 4

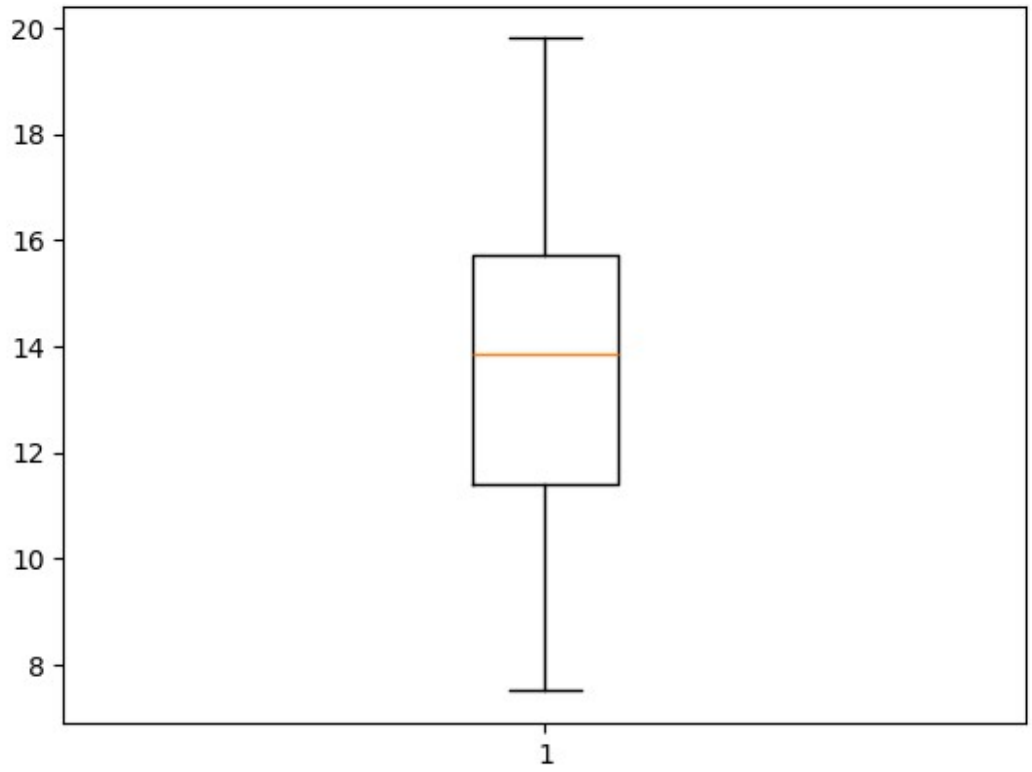
Viene effettuato un test di rottura di un certo materiale ottenendo i seguenti valori in megapascal (MPa).

19.8	10.1	14.9	7.5	15.4	15.4
15.4	18.5	7.9	12.7	11.9	11.4
11.4	14.1	17.6	16.7	15.8	
19.5	8.8	13.6	11.9	11.4	

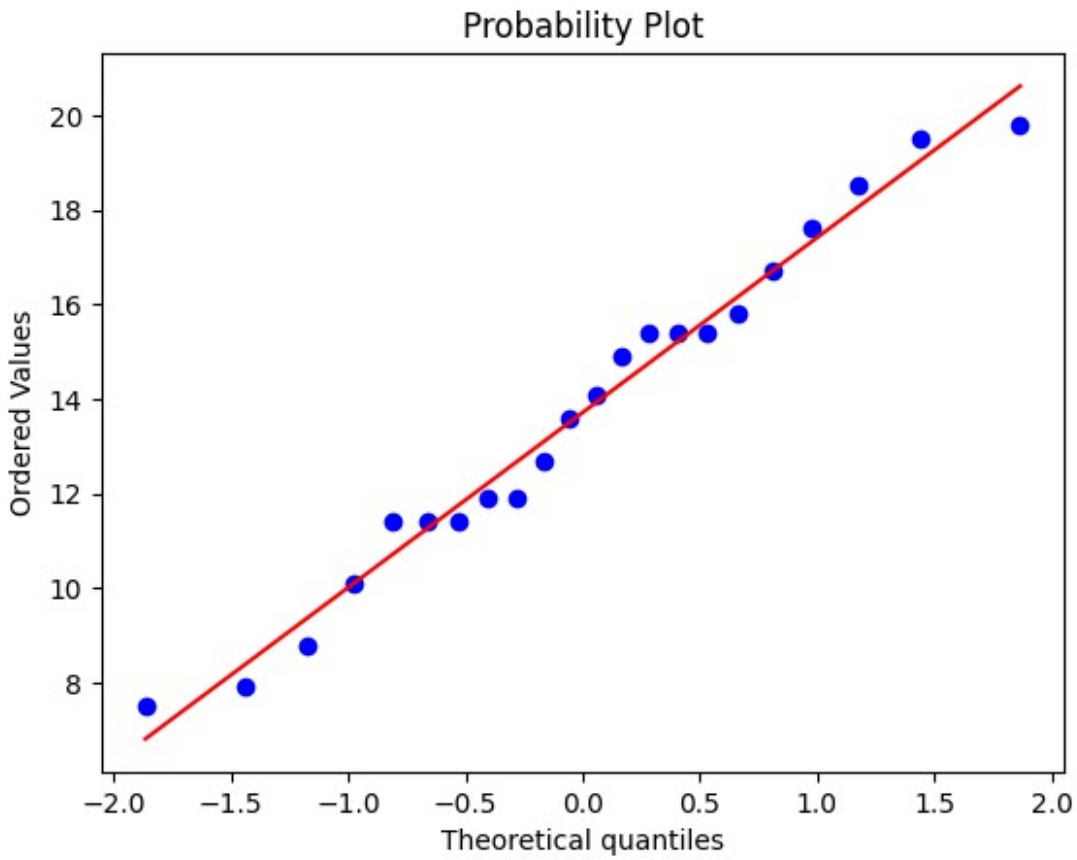
Dopo aver verificato graficamente che il campione proviene da una popolazione distribuita approssimativamente in modo normale, determinare l'intervallo di confidenza al 95% per la media

```
import numpy as np
from scipy.stats import t
import matplotlib.pyplot as plt

X = np.array(
```

```
    [
        19.8,
        10.1,
        14.9,
        7.5,
        15.4,
        15.4,
        15.4,
        18.5,
        7.9,
        12.7,
        11.9,
        11.4,
        11.4,
        14.1,
        17.6,
        16.7,
        15.8,
        19.5,
        8.8,
        13.6,
        11.9,
        11.4,
    ]
)

# Verifichiamo che il campione provenga da una distribuzione
approssimativamente normale

media = np.mean(X)  # Media
S = np.std(X, ddof=1)  # Restituisce la deviazione standard

plt.boxplot(X)
plt.show()
```




```
# Grafico di probabilità normale

from scipy.stats import probplot
from scipy.stats import norm

probplot(X, dist=norm, plot=plt)
plt.show()
```




```
n = X.size
gradiliberta = n - 1
alpha = 0.05

t = t.ppf(1 - alpha / 2, gradiliberta)

min = media - S / np.sqrt(n) * t
max = media + S / np.sqrt(n) * t

print("Intervallo di confidenza [{}, {}]".format(min, max))
```

```
Intervallo di confidenza [12.138069152904343, 15.289203574368383]
```

Esercizio 5

Un macchinario riempie automaticamente delle bottiglie. Da un campione di 20 misurazioni si ottengono i seguenti valori (in litri)

2.05, 2.04, 1.98, 1.96, 2.03, 2.01, 1.97, 1.99, 2.01, 2.05
1.96, 1.95, 2.04, 2.01, 1.97, 1.96, 2.02, 2.04, 1.98, 1.94

Se la varianza fosse troppo grande, la proporzione di bottiglie sotto o sovrariempite sarebbe non accettabile.

Calcolare l'intervallo di confidenza al 95% per il limite superiore per la deviazione standard.

```
import numpy as np
from scipy.stats import chi2

X = np.array(
    [
        2.05,
        2.04,
        1.98,
        1.96,
        2.03,
        2.01,
        1.97,
        1.99,
        2.01,
        2.05,
        1.96,
        1.95,
        2.04,
        2.01,
        1.97,
        1.96,
        2.02,
        2.04,
        1.98,
        1.94,
    ]
)
n = X.size

S2 = np.std(X, ddof=1) ** 2

alpha = 0.05
chi2 = chi2.ppf(1 - alpha, n)

SIG2 = S2 * (n - 1) / chi2

print("Limite superiore per la varianza: ", SIG2)

SIG = np.sqrt(SIG2)

print("Limite superiore per la deviazione standard: ", SIG)
```

```
Limite superiore per la varianza:  0.0007933669721643767
Limite superiore per la deviazione standard:  0.02816677070884017
```

Esercizi

March 27, 2024

1 Esercizio 1

Le specifiche tecniche per la velocità di combustione di un propellente richiedono che essa sia di 50 cm/s. Sappiamo che la deviazione standard è di 2 cm/s. Si effettuano 24 misurazioni: 51.0, 50.2, 49.5, 48.7, 50.2, 50.5, 49.6, 51.1, 50.6, 49.1, 53.1, 50.4, 49.3, 48.9, 50.3, 51.8, 51.3, 48.5, 49.3, 55.1, 53.1, 52.5, 55.1, 50.6.

Si può rigettare l'ipotesi nulla con un livello di significatività del 5%? E se invece si richiedesse l'1%? Calcolare infine il p-value.

Supponiamo che lo sperimentatore voglia impostare il test in modo che la reale velocità di combustione media differisca da 50 cm/s per al più 1 cm/s. Si vuole inoltre che il test affermerà questo fatto (cioè rigetterà H0 : $\mu = 50$) con una probabilità del 90% e un livello di significatività del 5%. Determinare la dimensione campionaria.

[91]:
```
import numpy as np
from scipy.stats import norm

X = np.array(
    [
        51.0,
        50.2,
        49.5,
        48.7,
        50.2,
        50.5,
        49.6,
        51.1,
        50.6,
        49.1,
        53.1,
        50.4,
        49.3,
        48.9,
        50.3,
        51.8,
        51.3,
        48.5,
        49.3,
```

```
        55.1,
        53.1,
        52.5,
        55.1,
        50.6,
    ]
)
n = X.size

sigma = 2
mu_0 = 50

media = np.mean(X)

Z_0 = (media - mu_0) / sigma * np.sqrt(n)

# Test con livello di significatività del 5%

alpha = 0.05
PHI = norm.ppf(1 - alpha / 2)

print("Z0: ", Z_0)
print("PHI: ", PHI)

print("Z0 > PHI: Si rigetta l'ipotesi nulla")
```

```
Z0:  2.020829037796111
PHI:  1.959963984540054
Z0 > PHI: Si rigetta l'ipotesi nulla
```

[92]:
```
# Test con livello di significatività del 1%

alpha = 0.01
PHI = norm.ppf(1 - alpha / 2)

print("Z0: ", Z_0)
print("PHI: ", PHI)

print("Z0 < PHI: Non abbiamo elementi sufficienti per rigettare l'ipotesi␣
 ↪nulla")
```

```
Z0:  2.020829037796111
PHI:  2.5758293035489004
Z0 < PHI: Non abbiamo elementi sufficienti per rigettare l'ipotesi nulla
```

[93]:
```
# Calcolo del p-value

p_value = 2 * (1 - norm.cdf(np.abs(Z_0)))
```

```
print("p-value: ", p_value)
```

```
p-value:  0.04329746577930438
```

[94]:
```
# Determiniamo la dimensione campionaria

alpha = 0.05
beta = 0.1
delta = 1

PHI = norm.ppf(1.0 - alpha / 2)
print(-PHI - delta * np.sqrt(n) / sigma, "< -3")

dimensione_campionaria = (PHI + norm.ppf(1 - beta)) ** 2 * sigma**2 / delta**2

print("Dimensione campionaria: ", dimensione_campionaria)
```

```
-4.409453727323232 < -3
Dimensione campionaria:  42.029692245762476
```

2 Esercizio 2

Si vuole testare ad un livello di significatività = 0.05 se il carico di rottura di un materiale supera 10 MPa, tenendo presente che 22 prove hanno fornito i seguenti risultati

19.8 18.5 17.6 16.7 15.8 15.4 14.1 13.6 11.9 11.4 11.4 8.8 7.5 15.4 15.4 19.5 14.9 12.7 11.9 11.4 10.1 7.9

Calcolare inoltre il p-value.

[95]:
```
import numpy as np
from scipy.stats import t

X = np.array(
    [
        19.8,
        18.5,
        17.6,
        16.7,
        15.8,
        15.4,
        14.1,
        13.6,
        11.9,
        11.4,
        11.4,
        8.8,
        7.5,
        15.4,
```

```
        15.4,
        19.5,
        14.9,
        12.7,
        11.9,
        11.4,
        10.1,
        7.9,
    ]
)
n = X.size

media = np.mean(X)
S = np.std(X, ddof=1)
mu_0 = 10

T_0 = (media - mu_0) / S * np.sqrt(n)

alpha = 0.05
t = t.ppf(1 - alpha, n - 1)

print("T0: ", T_0)
print("t: ", t)

print("T0 > T: Si può rigettare l'ipotesi")
```

```
T0:  4.901682101212391
t:  1.7207429028118775
T0 > T: Si può rigettare l'ipotesi
```

[96]:
```
# Calcolo del p-value
from scipy.stats import t



p_value = 1 - t.cdf(T_0, n - 1)
print("p-value: ", p_value)
```

```
p-value:  3.781272593450513e-05
```

3 Esercizio 3

Un macchinario riempie automaticamente delle bottiglie. Da un campione di 20 misurazioni si ottengono i seguenti valori (in litri)

2.05, 2.04, 1.98, 1.96, 2.03, 2.01, 1.97, 1.99, 2.01, 2.05 1.96, 1.95, 2.04, 2.01, 1.97, 1.96, 2.02, 2.04, 1.98, 1.94

Se la deviazione standard fosse superiore a 0.05 litri, la proporzione di bottiglie sotto o sovrariempite

sarebbe non accettabile.

I dati del campione contengono prove che suggeriscono che il produttore abbia un problema con le bottiglie riempite troppo o troppo poco? Utilizzare = 0.05 e assumere che il volume di riempimento abbia una distribuzione normale.

[97]:
```
import numpy as np
from scipy.stats import chi2

# Test unilatero a sinistra

X = np.array(
    [
        2.05,
        2.04,
        1.98,
        1.96,
        2.03,
        2.01,
        1.97,
        1.99,
        2.01,
        2.05,
        1.96,
        1.95,
        2.04,
        2.01,
        1.97,
        1.96,
        2.02,
        2.04,
        1.98,
        1.94,
    ]
)
n = X.size

sig0 = 0.05
S = np.std(X, ddof=1)

W0 = S**2 / sig0**2 * (n - 1)

alpha = 0.05
CHI = chi2.ppf(alpha, n - 1)

print("W0: ", W0)
print("CHI: ", CHI)
```

```
print("W0 < CHI: Si può rigettare l'ipotesi")
```

```
W0:  9.967999999999988
CHI:  10.117013063859044
W0 < CHI: Si può rigettare l'ipotesi
```

4 Esercizio 4

In una catena di produzione si vuole mantenere il numero di pezzi difettosi al di sotto del 5%.

Si analizza un campione di 200 pezzi e si trovano 4 pezzi difettosi. Si può asserire ad un livello di significativit'a = 0.05 che la produzione rispetta le aspettative?

Supponendo che il valore vero sia p∗ = 0.03 e supponendo che il costruttore voglia accettare un valore dell'errore di secondo tipo = 0.1, quale ampiezza dovrebbe avere il campione?

[98]:
```
import numpy as np
from scipy.stats import norm

# Test unilatero a sinistra

p0 = 0.05
n = 200
p = 4 / n

Z_0 = (p - p0) / (np.sqrt(p0 * (1 - p0))) * np.sqrt(n)

alpha = 0.05
PHI = norm.ppf(alpha)

print("Z0: ", Z_0)
print("PHI: ", PHI)

print("Z0 < PHI: Si può rigettare l'ipotesi")
```

```
Z0:  -1.946657053569151
PHI:  -1.6448536269514729
Z0 < PHI: Si può rigettare l'ipotesi
```

[99]:
```
p_star = 0.03
beta = 0.1

dimensione_campionaria = (
    (norm.ppf(beta) * np.sqrt(p_star * (1.0 - p_star)) + PHI * np.sqrt(p0 * (1.
 ↪0 - p0)))
    / (p0 - p_star)
) ** 2.0
print("Dimensione campionaria: ", dimensione_campionaria)
```

```
Dimensione campionaria:  832.6221546780524
```

Si desidera confrontare due tipi di preparati per pittura, presumendo che abbiano tempi di essiccamento diversi. Si ipotizza che la deviazione standard del tempo di essiccamento per ciascun tipo di preparato sia di 8 minuti. Sono state tinteggiate 10 pareti con il trattamento 1 e altrettante con il trattamento 2. Le medie campionarie rilevate sono $X^- = 121$ minuti e $Y^- = 112$ minuti. Si intende stabilire se il tempo di essiccamento del campione 1 sia maggiore di quello del campione 2, con un livello di significatività $= 0.05$.

Inoltre, si procede al calcolo dell'intervallo di confidenza per la differenza dei tempi medi di essiccamento.

[100]:
```
import numpy as np
from scipy.stats import norm

# Test unilatero a destra

X_1 = 121
X_2 = 112
n1 = n2 = 10
sig1 = sig2 = 8
alpha = 0.05

Z_0 = (X_1 - X_2) / np.sqrt(sig1**2 / n1 + sig2**2 / n2)
PHI = norm.ppf(1 - alpha)

print("Z0: ", Z_0)
print("PHI: ", PHI)

print("Z0 > PHI: Si può rigettare l'ipotesi")
```

```
Z0:  2.5155764746872635
PHI:  1.6448536269514722
Z0 > PHI: Si può rigettare l'ipotesi
```

[101]:
```
# Intervallo di confidenza per la differenza dei tempi medi di essiccamento

intervallo = X_1 - X_2 - PHI * np.sqrt(sig1**2 / n1 + sig2**2 / n2)
print("Intervallo di confidenza: ", intervallo)
```

```
Intervallo di confidenza:  3.115192763359085
```

Quindici adulti, con età compresa tra 35 e 50 anni, hanno partecipato a uno studio per valutare gli effetti di dieta alimentare ed esercizio fisico sul livello di colesterolo nel sangue. In ciascun individuo, il livello di colesterolo è stato misurato inizialmente e tre mesi dopo l'adozione della dieta e dell'allenamento. Con un livello di significatività $= 0.05$, si desidera stabilire se la dieta ed esercizio fisico portino effettivamente a una riduzione del livello medio di colesterolo.

Blood Cholesterol Level		
Subject	Before	After
1	265	229
2	240	231
3	258	227
4	295	240
5	251	238
6	245	241
7	287	234
8	314	256
9	260	247
10	279	239
11	283	246
12	240	218
13	238	219
14	225	226
15	247	233

[102]:
```
import numpy as np
from scipy.stats import t

# Test unilatero a destra

X = np.array(
    [
        [
            265.0,
            240.0,
            258.0,
            295.0,
            251.0,
            245.0,
            287.0,
            314.0,
            260.0,
            279.0,
            283.0,
            240.0,
            238.0,
            225.0,
            247.0,
        ],
        [
            229.0,
            231.0,
            227.0,
            240.0,
            238.0,
            241.0,
            234.0,
            256.0,
```

```
            247.0,
            239.0,
            246.0,
            218.0,
            219.0,
            226.0,
            233.0,
        ],
    ]
)

D = X[0, :] - X[1, :]
n = D.size

D_bar = np.mean(D)
S = np.std(D, ddof=1)

T_0 = D_bar / S * np.sqrt(n)

alpha = 0.05
t = t.ppf(1 - alpha, n - 1)

print("T0: ", T_0)
print("T: ", t)

print("T0 > T: Si può rigettare l'ipotesi")
```

```
T0:  5.4658739941050065
T:  1.7613101357748562
T0 > T: Si può rigettare l'ipotesi
```

Si vuole testare se un dado sia equilibrato o meno ad un livello di significatività $= 0.05$. Si effettuano 100 lanci, registrando i risultati riportati nella seguente tabella.

faccia	1	2	3	4	5	6
N_i	20	7	12	18	20	23

Dopo aver aumentato la dimensione campionaria, si registrano i risultati riportati nella seguente tabella. Si ripeta il test di cui sopra con i nuovi dati.

faccia	1	2	3	4	5	6
N_i	388	322	314	316	344	316

Si ripeta quest'ultimo test con = 0.01.

[103]:
```
import numpy as np
from scipy.stats import chi2

X = np.array([20.0, 7.0, 12.0, 18.0, 20.0, 23.0])
N = np.sum(X)
p = 1 / 6

E = np.ones(6) * p * N

T = np.sum((X - E) ** 2.0 / E)

alpha = 0.05
CHI = chi2.ppf(1.0 - alpha, 6 - 1)

print("T: ", T)
print("CHI: ", CHI)

print("T < CHI: NON è possibile rigettare l'ipotesi")
```

```
T:  10.760000000000002
CHI:  11.070497693516351
T < CHI: NON è possibile rigettare l'ipotesi
```

[104]:
```
Y = np.array([388.0, 322.0, 314.0, 316.0, 344.0, 316.0])
N = np.sum(Y)
p = 1 / 6

E = np.ones(6) * p * N
T = np.sum((Y - E) ** 2.0 / E)

print("T: ", T)
print("CHI: ", CHI)

print("T > CHI: Si può rigettare l'ipotesi")
```

```
T:  12.616000000000001
CHI:  11.070497693516351
T > CHI: Si può rigettare l'ipotesi
```

Si può adattare una distribuzione di Poisson ai dati della seguente tabella?

x_i	0	1	2	3	4
N_i	584	398	165	35	15

[105]:
```
import numpy as np
from scipy.stats import chi2
from scipy.stats import poisson

X = np.array([584.0, 398.0, 165.0, 35.0, 15.0])
m = X.size
N = np.sum(X)

p = X / N
X_n = X * np.array([0, 1, 2, 3, 4])
lam = sum(X_n) / N

p0 = np.zeros(5)
for i in range(4):
    p0[i] = poisson.pmf(i, lam)
p0[4] = 1.0 - np.sum(p0)

T = N * np.sum((p - p0) ** 2.0 / p0)
print("T: ", T)

alpha = 0.05
CHI = chi2.ppf(1.0 - alpha, m - 1 - 1)
print("CHI: ", CHI)
```

```
T:  7.6054880723053175
CHI:  7.814727903251179
```

Si vuole testare se un antibiotico è efficace. Si considerano 170 pazienti. I dati ottenuti sono stati raccolti nella tabella seguente, detta tabella di contingenza

	pazienti guariti	pazienti non guariti	
pazienti non trattati	44	10	54
pazienti trattati	81	35	116
	125	45	

I due effetti, trattamento e guarigione, sono indipendenti? Si usi = 0.05

[106]:
```
import numpy as np
from scipy.stats import chi2

X = np.array([[44., 10.],[81., 35]])
n = 170

p = np.array([54., 116.])/n
q = np.array([125., 45.])/n
```

```
# Probabilità congiunta empirica
pi = X/n

T = 0

for h in range(2):
    for k in range(2):
        T = T + (p[h]*q[k]-pi[h][k])**2./pi[h][k]
T = n*T

alpha = 0.05
CHI = chi2.ppf(1.-alpha,(2-1)*(2-1))

print("T: ", T)
print("CHI: ", CHI)

print("T < CHI: NON è possibile rigettare l'ipotesi")
```

```
T:  3.0175117739708286
CHI:  3.841458820694124
T < CHI: NON è possibile rigettare l'ipotesi
```

5 Esercizio

Un motore a reazione è formato legando insieme un propellente di accensione e un propellente di sostegno all'interno di un alloggiamento metallico. La resistenza al taglio del legame tra i due tipi di propellente è una caratteristica importante. Vogliamo testare l'ipotesi che la mediana della resistenza al taglio sia 2000 psi con una significatività = 0.05. I dati sono riportati nel file Dataset motore.dat.

[107]:
```
#test bilatero
import numpy as np
from scipy.stats import binom
```

[108]:
```
alpha=0.05
mediana_0 = 2000
```

[109]:
```
X = np.loadtxt('Dataset_motore.dat')
display(X)
n=X.size
```

```
array([2158.7 , 1678.15, 2316.  , 2061.3 , 2207.5 , 1708.3 , 1784.7 ,
       2575.1 , 2357.9 , 2256.7 , 2165.2 , 2399.55, 1779.8 , 2336.75,
       1765.3 , 2053.5 , 2414.4 , 2200.5 , 2654.2 , 1753.7 ])
```

[110]:
```
#Calcoliamo le differenze
D=X-mediana_0
print(D)
```

```
[ 158.7  -321.85  316.     61.3   207.5  -291.7  -215.3   575.1   357.9
  256.7   165.2   399.55 -220.2   336.75 -234.7    53.5   414.4   200.5
  654.2  -246.3 ]
```

[111]:
```
r_piu = D>0 #Diventa un array di bool -> true se la condizione (D>0) è vera
r_piu = r_piu.astype(int) #Conversione in int (quindi diventeranno tutti 0 o 1)
r_piu = np.sum(r_piu) #somma dei valori >0
```

[113]:
```
print("r+: ", r_piu)
print("n/2: ", n/2)
print("r+ > n/2")
k=range(r_piu,n+1)
Y=binom.pmf(k,n,0.5)
p_value=2*(np.sum(Y))

print("p_value ",p_value)
# Poiché il p-value è maggiore della soglia fissata alpha = 0.05
# Non abbiamo elementi sufficienti per rigettare H0
```

```
r+:  14
n/2:  10.0
r+ > n/2
p_value  0.11531829833984363
```

6 Esercizio

In un esperimento si confrontano quattro diverse tecniche di mescolamento per il cemento e si misura la resistenza alla trazione. Si può affermare che la tecnica di mescolamento influisca sulla resistenza alla trazione? Si usi = 0.05.

Mixing Technique	Tensile Strength (lb/in.^{2})			
1	3129	3000	2865	2890
2	3200	3000	2975	3150
3	2800	2900	2985	3050
4	2600	2700	2600	2765

[114]:
```
import numpy as np
```

[115]:
```
X_1 = np.array([3129., 3000., 2865., 2890.])
X_2 = np.array([3200., 3000., 2975., 3150.])
```

```
X_3 = np.array([2800., 2900., 2985., 3050.])
X_4 = np.array([2600., 2700., 2600., 2765.])
```

[117]:
```
alpha = 0.05
n = np.array([X_1.size, X_2.size, X_3.size, X_4.size])
m = 4
```

[118]:
```
N = np.sum(n)
Y = np.concatenate([X_1, X_2, X_3, X_4])
```

[119]:
```
from scipy.stats import rankdata
R = rankdata(Y)
display(R)
```

```
array([14. , 11.5,  6. ,  7. , 16. , 11.5,  9. , 15. ,  5. ,  8. , 10. ,
       13. ,  1.5,  3. ,  1.5,  4. ])
```

[120]:
```
S2 = (np.sum(R**2.)-N*(N+1)**2./4.) / (N-1)
display(S2)
```

```
22.6
```

[121]:
```
RR = np.zeros(m)
for i in range(m):
    RR[i] = np.sum(R[m*i:m*(i+1)])
display(RR)
```

```
array([38.5, 51.5, 36. , 10. ])
```

[122]:
```
H = (np.sum(RR**2. / n)-N*(N+1)**2./4.)/S2
display(H)
```

```
10.027654867256636
```

[123]:
```
from scipy.stats import chi2
alpha = 0.05
CHI = chi2.ppf(alpha, m-1)
display(CHI)
```

```
0.35184631774927144
```

[124]:
```
# Si rigetta H0 in favore di H1
```

Esercizi

March 27, 2024

1 Esercizio 1

Il peso corporeo e la pressione sistolica del sangue di 26 individui maschi selezionati in modo casuale nella fascia d'età che va da 25 a 30 anni sono mostrati in tabella. Assumiamo che il peso e la pressione sanguigna siano normalmente distribuiti.

1. Si determini la retta di regressione.
2. Si calcolino gli intervalli di confidenza per i coefficienti di regressione.
3. Si testi la significatività della regressione usando = 0.05.
4. Si calcoli il coefficiente di determinazione.

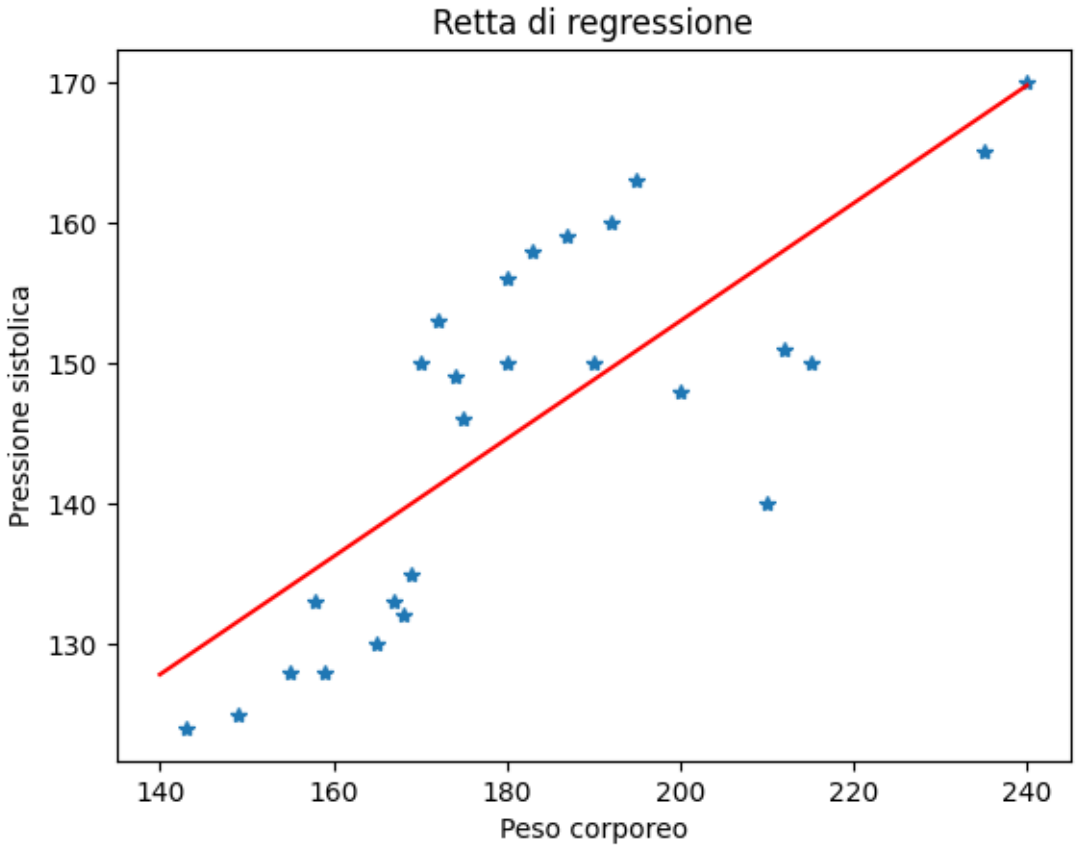
Subject	Weight	Systolic BP	Subject	Weight	Systolic BP
1	165	130	14	172	153
2	167	133	15	159	128
3	180	150	16	168	132
4	155	128	17	174	149
5	212	151	18	183	158
6	175	146	19	215	150
7	190	150	20	195	163
8	210	140	21	180	156
9	200	148	22	143	124
10	149	125	23	240	170
11	158	133	24	235	165
12	169	135	25	192	160
13	170	150	26	187	159

[349]:
```
import numpy as np

DATA = np.loadtxt("DATA_reg_lin.dat")
x = DATA[:, 1]  # Peso corporeo
y = DATA[:, 2]  # Pressione sistolica

# Calcolo dei coefficienti di regressione della retta y = b_0 + b_1*x

x_bar, y_bar = np.mean(x), np.mean(y)
```

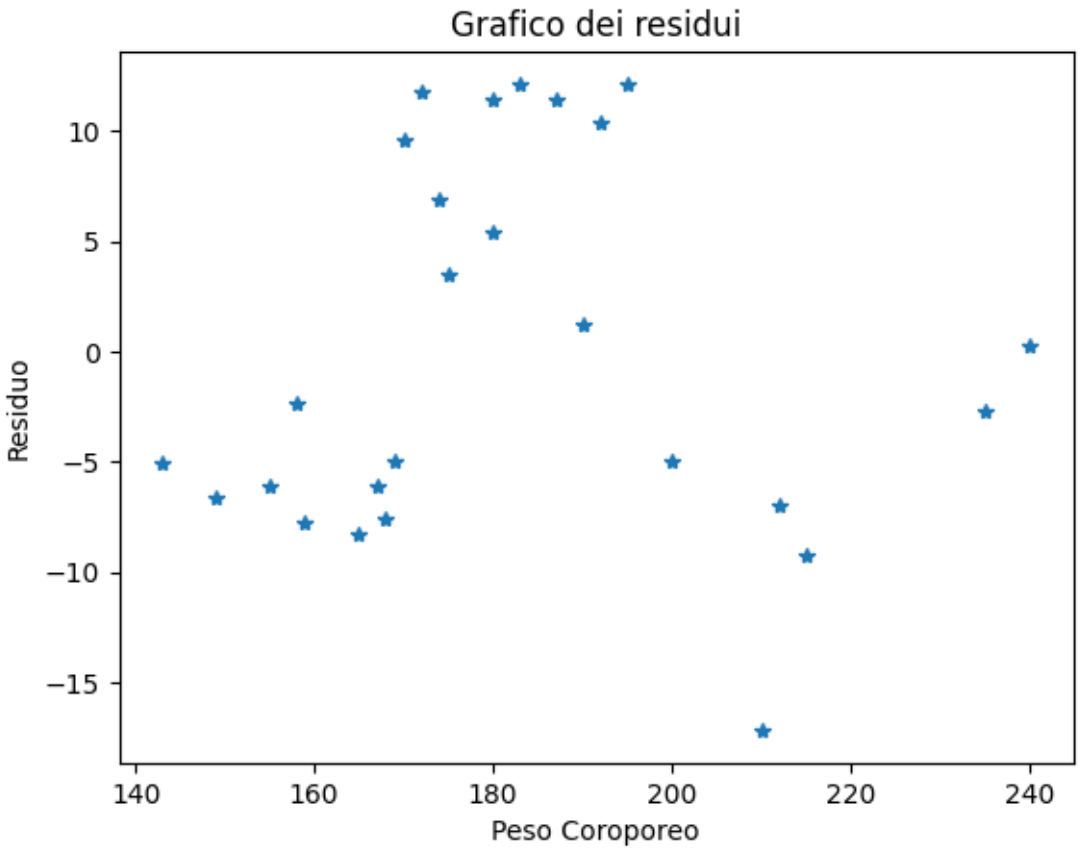
```
n = x.size

sig_xy = np.sum((x - x_bar) * (y - y_bar)) / n
sig_x_2 = np.sum((x - x_bar) ** 2) / n

b_0 = y_bar - sig_xy / sig_x_2 * x_bar
b_1 = sig_xy / sig_x_2

print("Coefficienti di regressione: ", b_0, b_1)
```

```
Coefficienti di regressione:  69.10437279118659 0.41941520291569645
```

[350]:
```
import matplotlib.pyplot as plt

assex = np.linspace(140, 240, 1000)
assey = b_0 + b_1 * assex

plt.plot(x, y, "*")
plt.plot(assex, assey, color="red")
plt.title("Retta di regressione")
plt.xlabel("Peso corporeo")
plt.ylabel("Pressione sistolica")
plt.show()
```



[351]:
```
# Calcolo dei residui

y_hat = b_0 + b_1 * x
r = y - y_hat

plt.plot(x, r, "*")
plt.title("Grafico dei residui")
plt.xlabel("Peso Coroporeo")
plt.ylabel("Residuo")
plt.show()
```

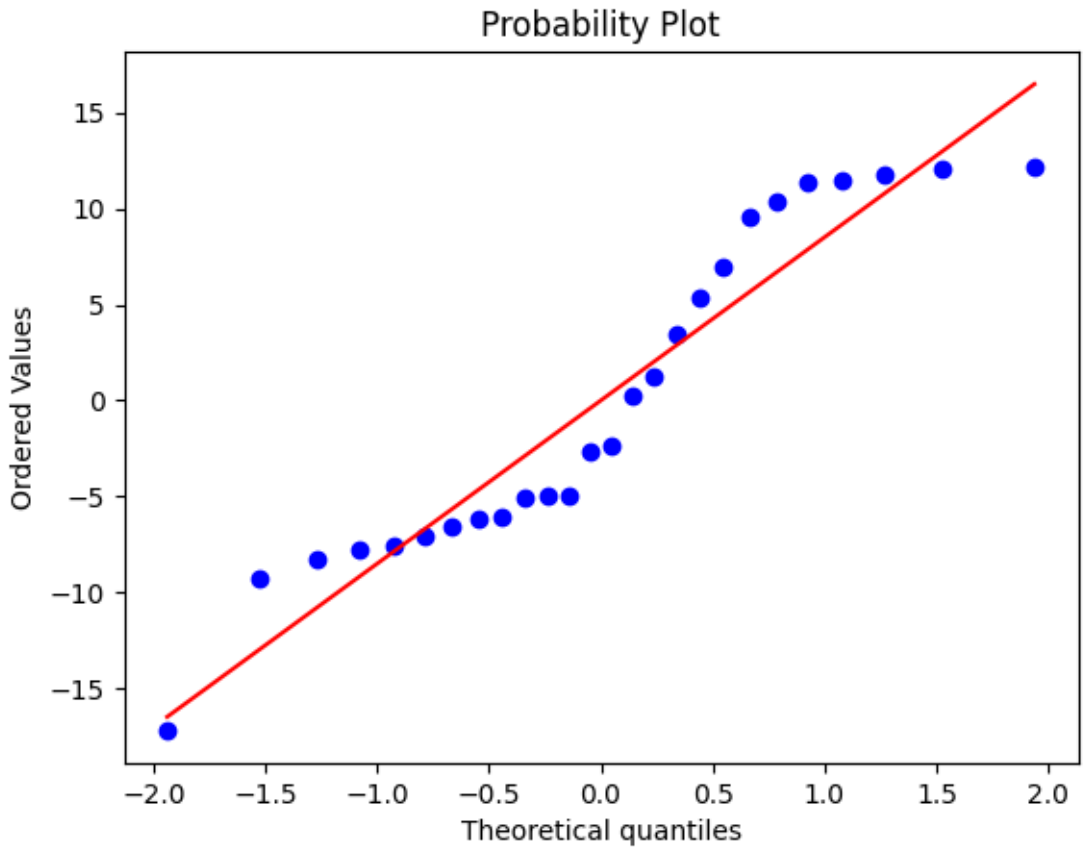


[352]:
```
# Calcolo degli intervalli di confidenza
from scipy.stats import t

s2 = np.sum(r**2) / (n - 2)
alpha = 0.05
```

```
T = t.ppf(1 - alpha / 2, n - 2)
s = np.sqrt(s2)
sig_x = np.sqrt(sig_x_2)

b_0_sx = b_0 - s * np.sqrt(1 / n + x_bar**2 / (n * sig_x_2)) * T
b_0_dx = b_0 + s * np.sqrt(1 / n + x_bar**2 / (n * sig_x_2)) * T

b_1_sx = b_1 - s / (sig_x * np.sqrt(n)) * T
b_1_dx = b_1 + s / (sig_x * np.sqrt(n)) * T

print("Intervallo di confidenza di b_0: [{},{}]".format(b_0_sx, b_0_dx))
print("Intervallo di confidenza di b_1: [{},{}]".format(b_1_sx, b_1_dx))
```

```
Intervallo di confidenza di b_0: [42.45917560573429,95.7495699766389]
Intervallo di confidenza di b_1: [0.27462811448021235,0.5642022913511806]
```

[353]:
```
from scipy.stats import probplot
from scipy.stats import norm

probplot(r, dist=norm, plot=plt)
plt.show()
```



[354]:
```
# Test di significatività

alpha = 0.05

T1 = np.abs(np.sqrt(n) * (b_1 / s) * sig_x)
t = t.ppf(1 - alpha / 2, n - 2)

print("T1: |{}|".format(T1))
print("t: ", t)
print("T1 >= t: Si rigetta l'ipotesi")
```

```
T1: |5.978643837487954|
t:  2.0638985616280205
T1 >= t: Si rigetta l'ipotesi
```

[355]:
```
# Calcolo del coefficiente di determinazione

sig_y_2 = np.sum((y - y_bar) ** 2) / n
R2 = sig_xy**2 / (sig_x_2 * sig_y_2)

print("Coefficiente di determinazione: ", R2)
```

```
Coefficiente di determinazione:  0.5982872450148403
```

2 Esercizio 2

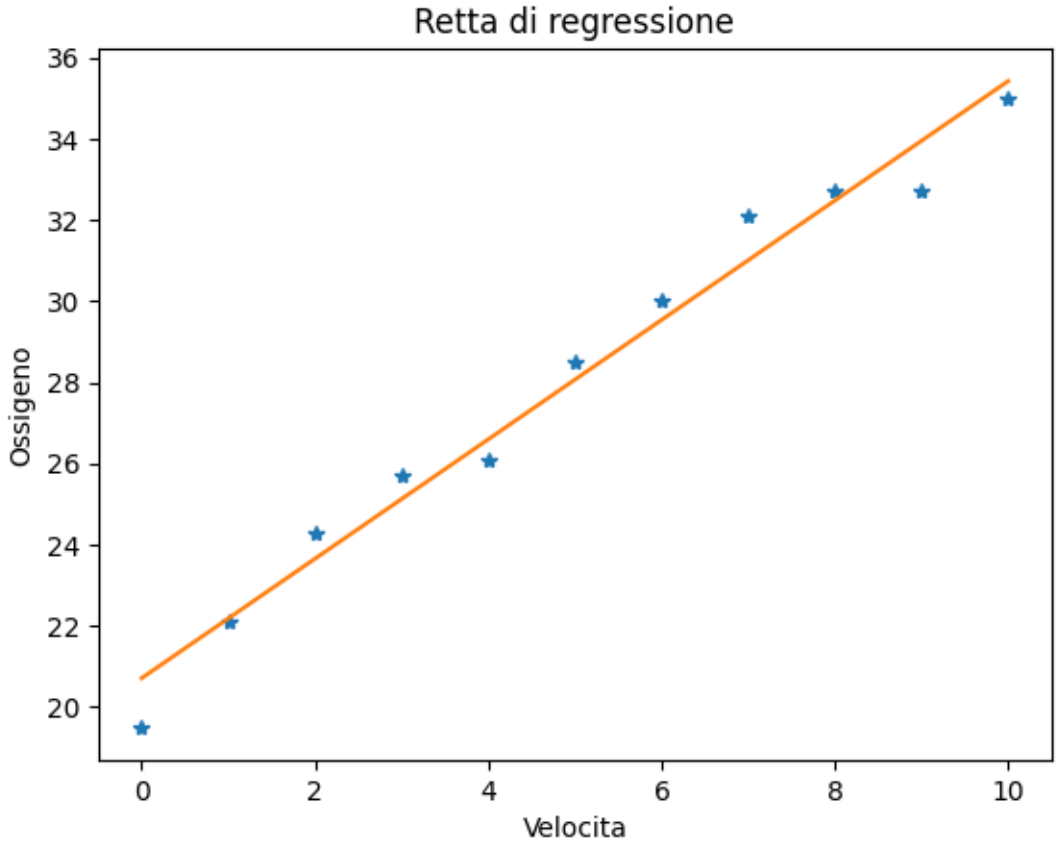
L'ossigeno consumato da una persona che cammina è funzione della sua velocità. La seguente tabella riporta il volume di ossigeno consumato a varie velocità di cammino.

Velocità (km/h)	Ossigeno (l/h)
0	19,5
1	22,1
2	24,3
3	25,7
4	26,1
5	28,5
6	30,0
7	32,1
8	32,7
9	32,7
10	35,0

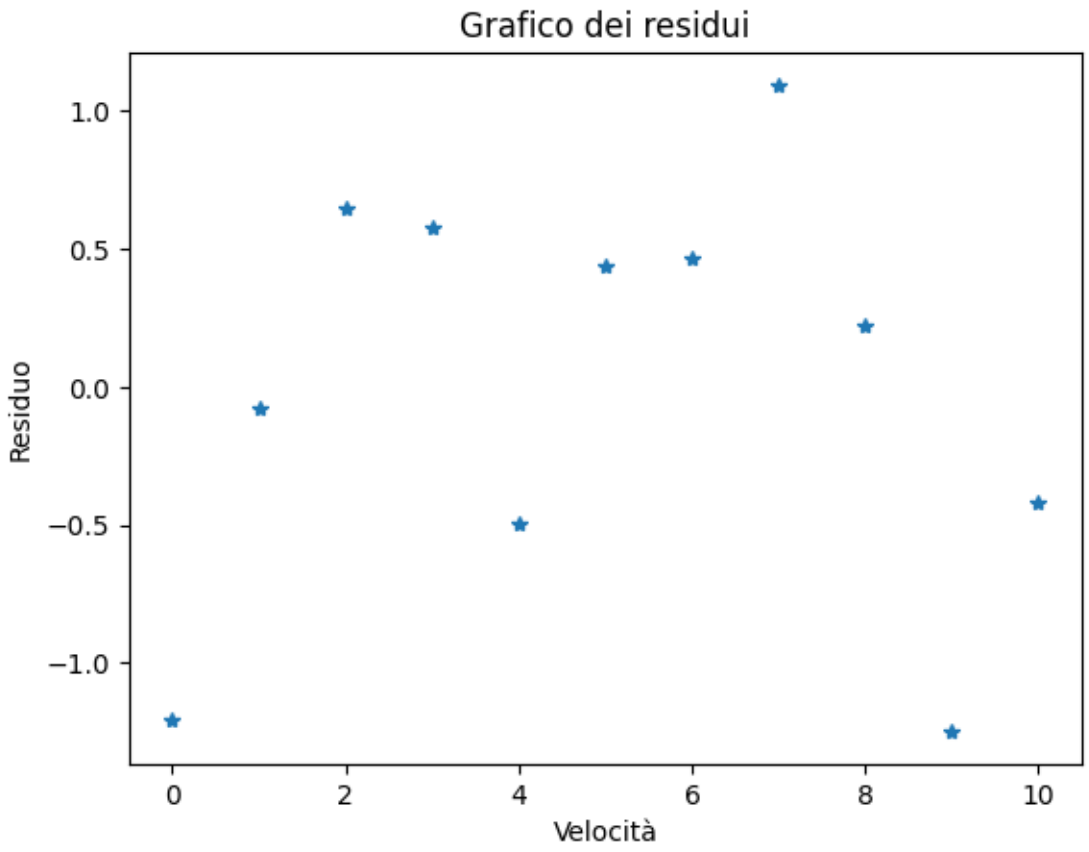
- Ipotizzando una relazione lineare, scrivere l'equazione della retta di regressione.
- Si testi la significatività della regressione usando $= 0.05$.

[356]:
```
import numpy as np
import matplotlib.pyplot as plt

X = np.arange(0, 11)
Y = np.array([19.5, 22.1, 24.3, 25.7, 26.1, 28.5, 30.0, 32.1, 32.7, 32.7, 35.0])

x_bar = np.mean(X)
y_bar = np.mean(Y)

n = X.size

sig_xy = np.sum((X - x_bar) * (Y - y_bar)) / n
sig_2_x = np.sum((X - x_bar) ** 2) / n

b0 = y_bar - ((sig_xy) / (sig_2_x)) * x_bar
b1 = sig_xy / sig_2_x

assex = np.linspace(0, 10, 100)
assey = b0 + b1 * assex
```

```
plt.plot(X, Y, "*")
plt.plot(assex, assey)
plt.title("Retta di regressione")
plt.xlabel("Velocita")
plt.ylabel("Ossigeno")
plt.show()
```



[357]:
```
# Calcoliamo i residui
y_hat = b0 + b1 * X
r = Y - y_hat

plt.plot(X, r, "*")
plt.title("Grafico dei residui")
plt.xlabel("Velocità")
plt.ylabel("Residuo")
plt.show()
s2 = np.sum(r**2) / (n - 2)
```



[358]:
```
# test di indipendenza
from scipy.stats import t

alpha = 0.05
s = np.sqrt(s2)
sig_x = np.sqrt(sig_2_x)

T1 = np.abs(np.sqrt(n) * (b1 / s) * sig_x)
t = t.ppf(1 - alpha / 2, n - 2)
print("T1: |{}|".format(T1))
print("t: ", t)
print("T1 >= t: Si rigetta l'ipotesi")
```

```
T1: |19.131004675087407|
t:  2.2621571627409915
T1 >= t: Si rigetta l'ipotesi
```

3 Esercizio 3

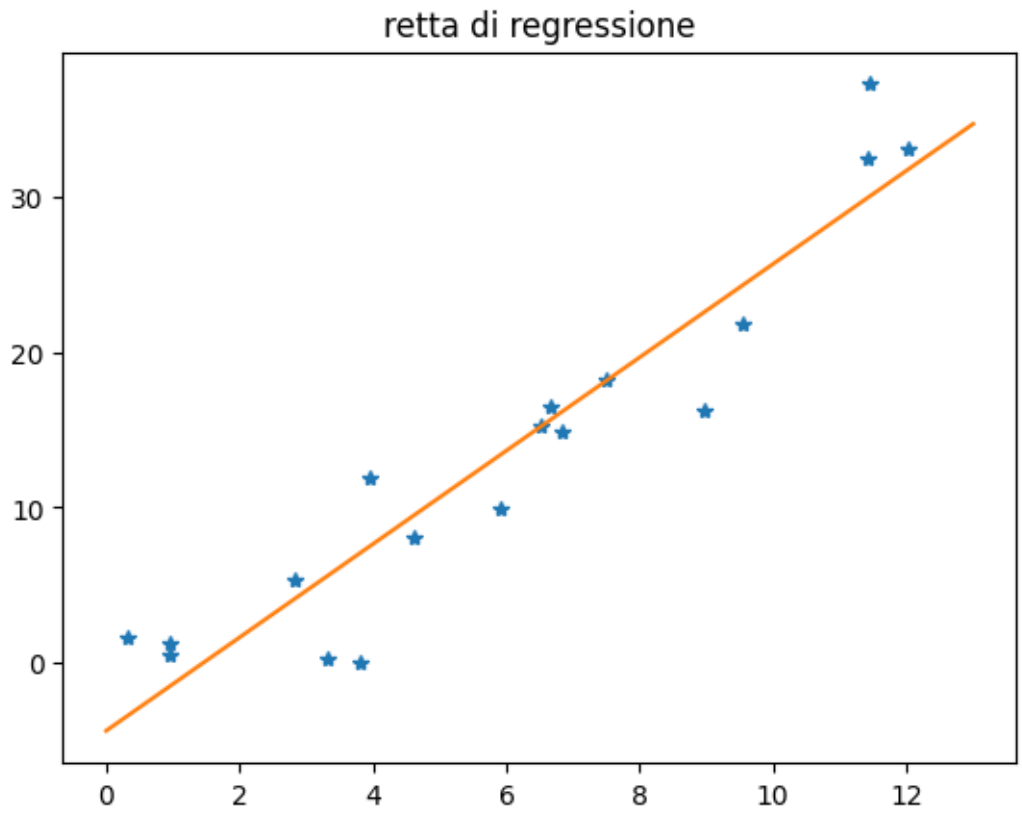
Consideriamo i dati di tabella. Formuliamo degli appropriati modelli di regressione.

x	y	x	y
0.32	1.60	6.65	16.47
2.83	5.30	8.96	16.25
3.94	11.88	11.45	37.30
6.52	15.28	0.96	0.48
7.51	18.19	3.80	0.06
11.43	32.55	5.91	9.91
0.96	1.23	6.83	14.96
3.32	0.32	9.56	21.82
4.62	8.03	12.02	33.19

.

[359]:
```
import numpy as np

DATA = np.loadtxt("DATA_reg_lin_2.dat")
x= DATA[:,0] #x
y= DATA[:,1] #y
```

[361]:
```
# Calcolo dei coefficienti di regressione y = b_0 + b_1 x
import matplotlib.pyplot as plt
n = x.size
x_bar = np.mean(x)
y_bar = np.mean(y)
sig_xy = np.sum((x-x_bar)*(y-y_bar))/n
sig_x_2 = np.sum((x-x_bar)**2.)/n

b_0 = y_bar - sig_xy/sig_x_2*x_bar
b_1 = sig_xy/sig_x_2

assex = np.linspace(0,13,100)
assey= b_0+b_1*assex
# Grafico non obbligatorio
plt.plot(x,y,"*")
plt.plot(assex,assey)
plt.title("retta di regressione")
plt.show()
```



[362]:
```
# Calcolo dei coefficienti di regressione usando le formule
# della regressione lineare multipla
```

[363]:
```
x1 = np.ones(n)
x2 = x
X = np.zeros((n,2))
X[:,0] = x1
X[:,1] = x2
display(X)
```

```
array([[ 1.  ,  0.32],
       [ 1.  ,  2.83],
       [ 1.  ,  3.94],
       [ 1.  ,  6.52],
       [ 1.  ,  7.51],
       [ 1.  , 11.43],
       [ 1.  ,  0.96],
       [ 1.  ,  3.32],
       [ 1.  ,  4.62],
       [ 1.  ,  6.65],
       [ 1.  ,  8.96],
```

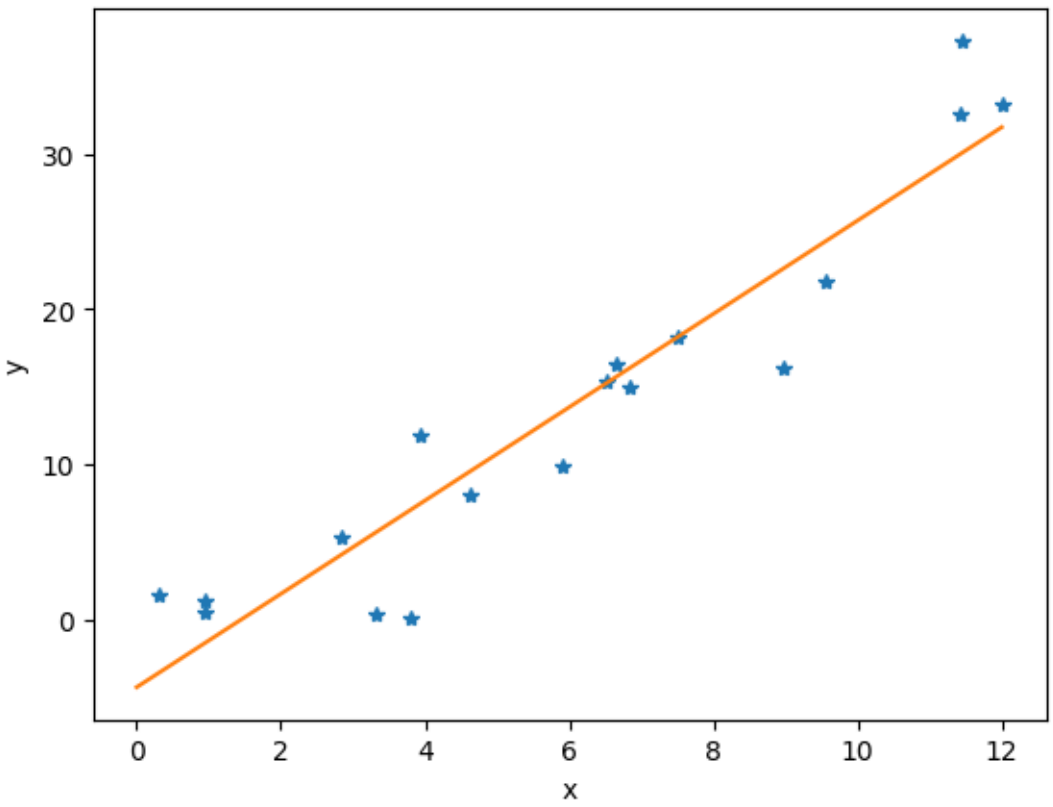
```
       [ 1.  , 11.45],
       [ 1.  ,  0.96],
       [ 1.  ,  3.8 ],
       [ 1.  ,  5.91],
       [ 1.  ,  6.83],
       [ 1.  ,  9.56],
       [ 1.  , 12.02]])
```

[364]:
```
import numpy as np
XX = np.linalg.pinv(X)
b = np.dot(XX,y)
display(b)
y_hat = np.dot(X,b)
display(y_hat)
```

```
array([-4.39133048,  3.01016775])
```

```
array([-3.42807679,  4.12744427,  7.46873047, 15.23496327, 18.21502935,
       30.01488694, -1.50156943,  5.60242646,  9.51564454, 15.62628508,
       22.57977259, 30.0750903 , -1.50156943,  7.04730699, 13.39876095,
       16.16811528, 24.38587324, 31.79088592])
```

[365]:
```
# Grafico
xx = np.linspace(0.,12.,100)
yy = b[0] + b[1]*xx

import matplotlib.pyplot as plt
plt.plot(x, y, '*')
plt.plot(xx, yy)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



[366]:
```
R2 = np.sum((y_hat-y_bar)**2.) / np.sum((y-y_bar)**2.)
display(R2)
```

0.8905996203945858

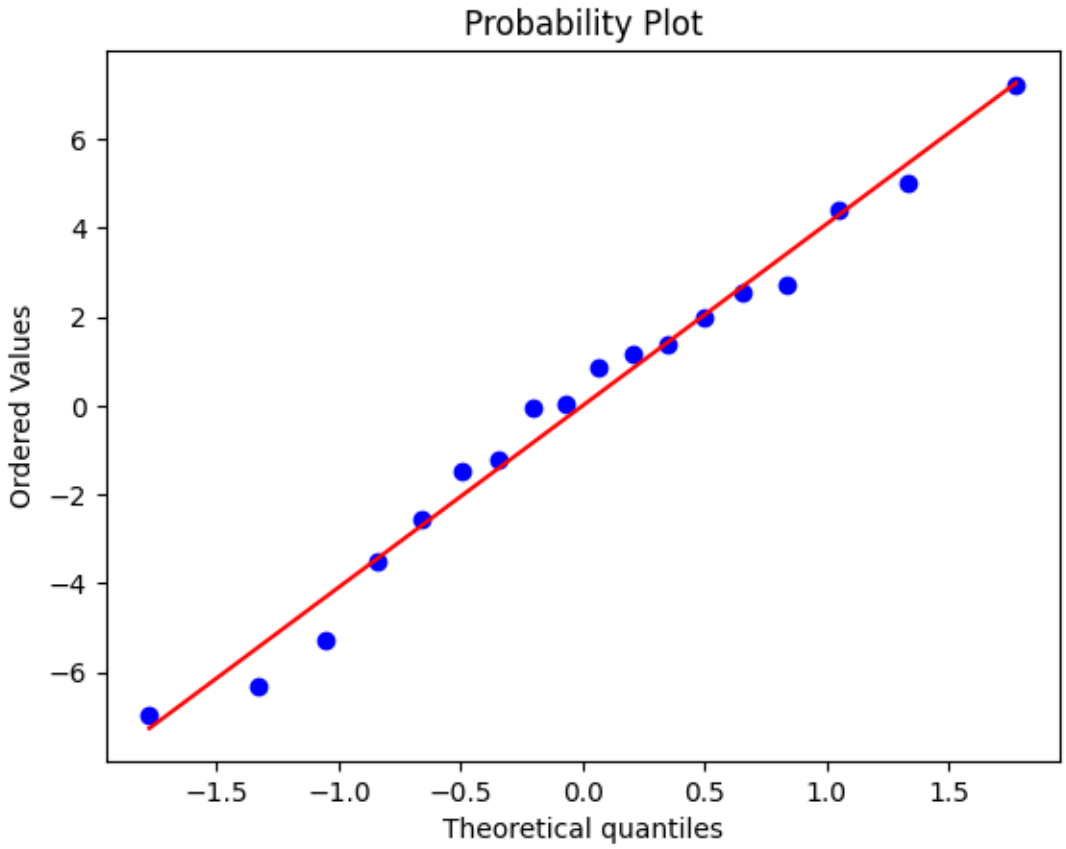
[367]:
```
# test di indipendenza
# H0 : b1 = 0
# H1 : b1 <> 0

M = np.linalg.inv(np.dot(X.T,X))
m = M[1][1]
r = y-y_hat
s2 = np.sum(r**2.)/(n-2)
T1 = b[1]/(np.sqrt(s2*m))
display(T1)

from scipy.stats import t
alpha = 0.05
tt = t.ppf(1.-alpha/2.,n-2)
display(tt)
```

11.412790093360774

```
2.119905299221011
```

[368]:
```
# Rigettiamo l'ipotesi nulla
# Pertanto vi è dipendenza di y dal predittore
```

[369]:
```
from scipy.stats import probplot
from scipy.stats import norm
fig, ax = plt.subplots(1, 1)
probplot(r, dist=norm, plot=ax)
plt.show()
```



[370]:
```
# Adottiamo un modello quadratico
X2 = np.zeros((n,3))
X2[:,0] = x1
X2[:,1] = x2
X2[:,2] = x2**2.
display(X2)
```

```
array([[1.000000e+00, 3.200000e-01, 1.024000e-01],
       [1.000000e+00, 2.830000e+00, 8.008900e+00],
       [1.000000e+00, 3.940000e+00, 1.552360e+01],
```

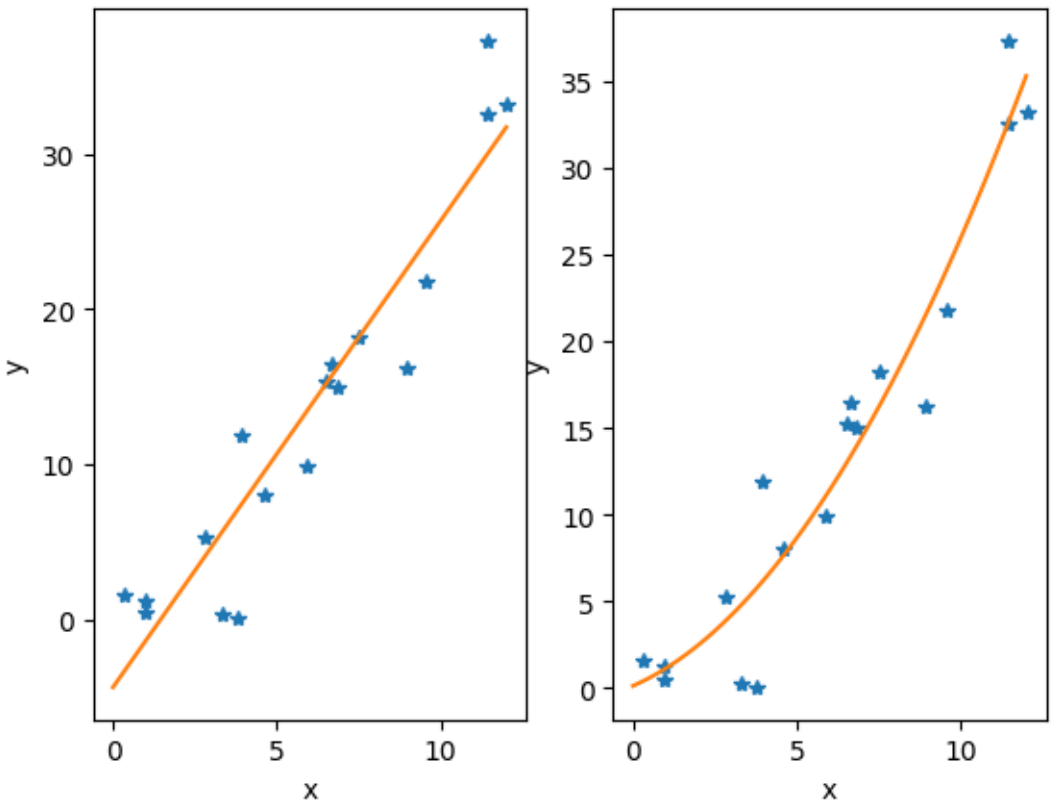
```
       [1.000000e+00, 6.520000e+00, 4.251040e+01],
       [1.000000e+00, 7.510000e+00, 5.640010e+01],
       [1.000000e+00, 1.143000e+01, 1.306449e+02],
       [1.000000e+00, 9.600000e-01, 9.216000e-01],
       [1.000000e+00, 3.320000e+00, 1.102240e+01],
       [1.000000e+00, 4.620000e+00, 2.134440e+01],
       [1.000000e+00, 6.650000e+00, 4.422250e+01],
       [1.000000e+00, 8.960000e+00, 8.028160e+01],
       [1.000000e+00, 1.145000e+01, 1.311025e+02],
       [1.000000e+00, 9.600000e-01, 9.216000e-01],
       [1.000000e+00, 3.800000e+00, 1.444000e+01],
       [1.000000e+00, 5.910000e+00, 3.492810e+01],
       [1.000000e+00, 6.830000e+00, 4.664890e+01],
       [1.000000e+00, 9.560000e+00, 9.139360e+01],
       [1.000000e+00, 1.202000e+01, 1.444804e+02]])
```

[371]:
```
XX2 = np.linalg.pinv(X2)
b2 = np.dot(XX2,y)
display(b2)
y_hat_2 = np.dot(X2,b2)
display(y_hat_2)
```

```
array([0.13714061, 0.83177463, 0.17471849])
```

```
array([ 0.42119966,  3.8903657 ,  6.12659254, 12.98766393, 16.23790817,
       32.47040378,  1.09666481,  4.82444942,  7.70920065, 13.39493015,
       21.61652091, 32.56699045,  1.09666481,  5.82081914, 11.15551343,
       13.96858652, 24.0570575 , 35.37846842])
```

[372]:
```
# Grafico
xx = np.linspace(0.,12.,100)
yy = b[0] + b[1]*xx
yy2 = b2[0] + b2[1]*xx + b2[2]*xx**2.

import matplotlib.pyplot as plt
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.plot(x, y, '*')
ax1.plot(xx, yy)
ax1.set_xlabel('x')
ax1.set_ylabel('y')

ax2.plot(x, y, '*')
ax2.plot(xx, yy2)
ax2.set_xlabel('x')
ax2.set_ylabel('y')
plt.show()
```



[373]:
```
M2 = np.linalg.inv(np.dot(X2.T,X2))
m2 = M2[1][1]
m3 = M2[2][2]
r2 = y-y_hat_2
s2_2 = np.sum(r2**2.)/(n-3)


# test di indipendenza
# H0 : b1 = 0
# H1 : b1 <> 0

T1_2 = b2[1]/(np.sqrt(s2_2*m2))
display(T1_2)

# test di indipendenza
# H0 : b2 = 0
# H1 : b2 <> 0

T2_2 = b2[2]/(np.sqrt(s2_2*m3))
display(T2_2)
```

```
from scipy.stats import t
tt = t.ppf(1.-alpha/2.,n-3)
display(tt)
```

1.0028596601848998

2.7266250601819872

2.131449545559323

[374]:
```
# Poiché T2_2 > tt si rigetta l'ipotesi nulla
# evidenziando una notevole dipendenza dal termine quadratico
```

[376]:
```
R2 = np.sum((y_hat_2-y_bar)**2.) / np.sum((y-y_bar)**2.)
display(R2)
```

0.9268534244836734

[ ]:
```
# Osserviamo che il coefficiente di determinazione è maggiore rispetto
# a quello del modello lineare suggerendo una migliore adeguatezza
# del modello quadratico
```

Esercizi

March 27, 2024

1 Esercizio 1

Scrivere un algoritmo per generare numeri pseudo casuali con distribuzione multinomiale B(1, 1/4, 1/2, 1/4). Generare 2000 di questi numeri e costruire un istogramma verificando l'accordo con la distribuzione teorica di probabilità

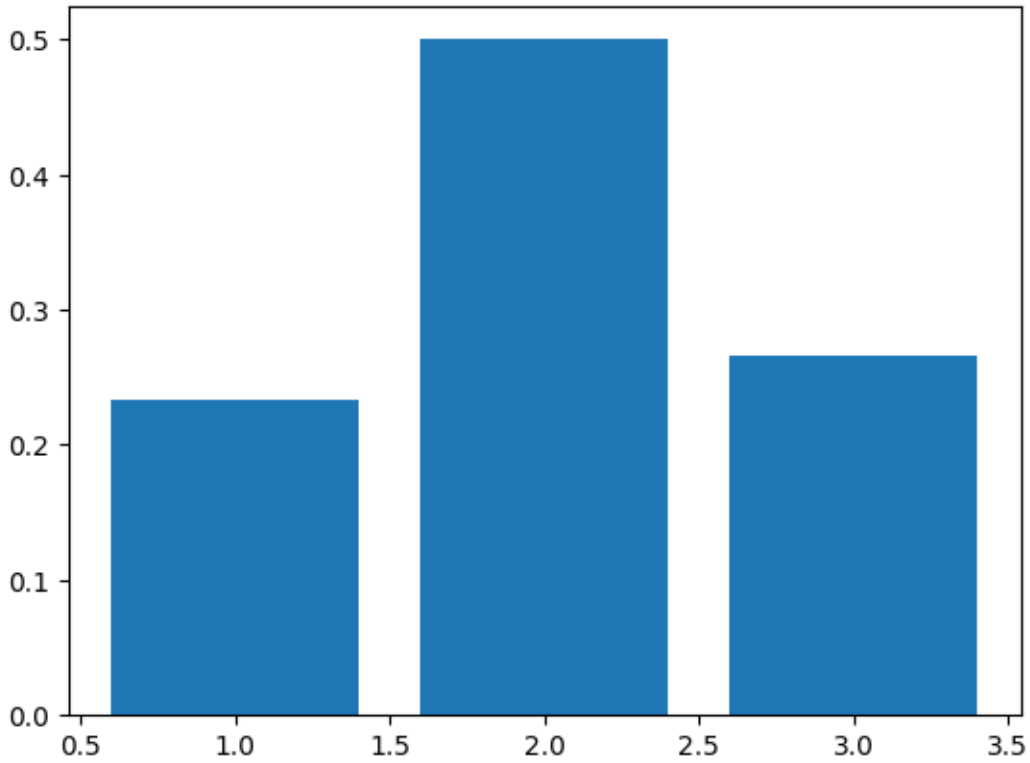
[225]:
```
import numpy as np
import matplotlib.pyplot as plt


def rand_multi(p):
    F = np.cumsum(p)
    xi = np.random.rand()
    X = np.nonzero(xi < F)[0][0] + 1
    return X
```

[226]:
```
p = np.array([1 / 4, 1 / 2, 1 / 4])

N = 2000
X = np.zeros(N)

for i in range(N):
    X[i] = rand_multi(p)

m = p.size
F = np.zeros(m)
for i in range(m):
    F[i] = np.sum(X == i + 1)

print(F)
print(np.sum(F))
```

```
[ 467. 1000.  533.]
2000.0
```

[227]:
```
x = np.arange(m) + 1
F = F / np.sum(F)
plt.bar(x, F)
```

```
plt.show()
```



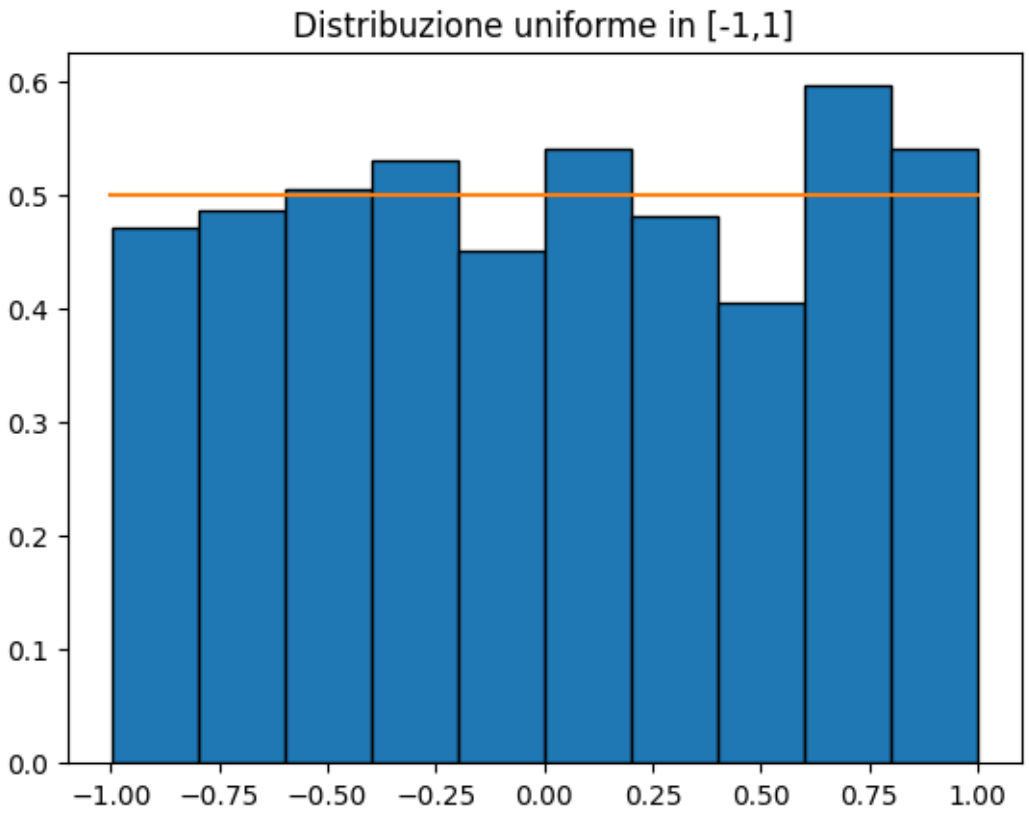
2 Esercizio 2

Si scrivano delle funzioni di Python per generare numeri random con distribuzione U([−1, 1]), EXP(3), N(1, 2), 2(5).

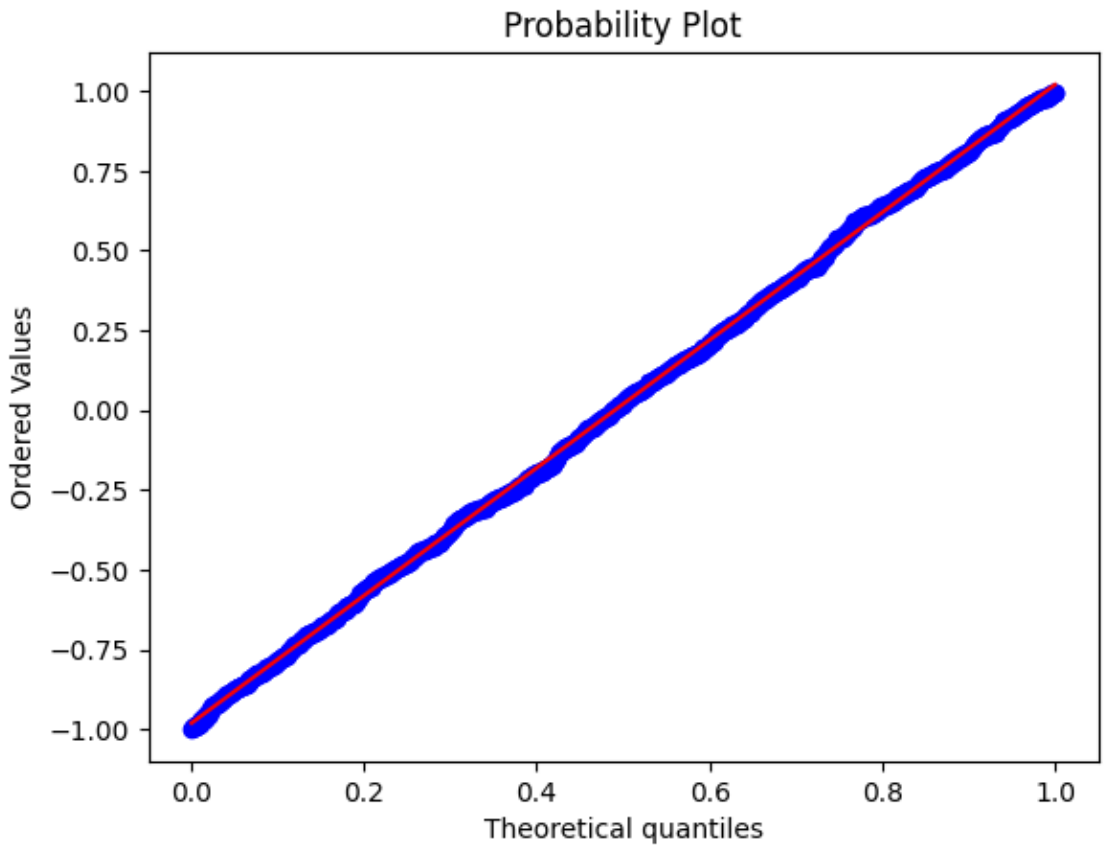
Si generino N = 1000 di questi numeri e si confrontino i risultati con le distribuzioni teoriche mediante istogramma e mediante grafico quantile-quantile.

[228]:
```
import numpy as np
import matplotlib.pyplot as plt

# Distribuzione uniforme U(-1,1)
from scipy.stats import uniform

a, b = -1, 1
N = 1000

X = np.random.rand(N)
Y = a + X * (b - a)
```

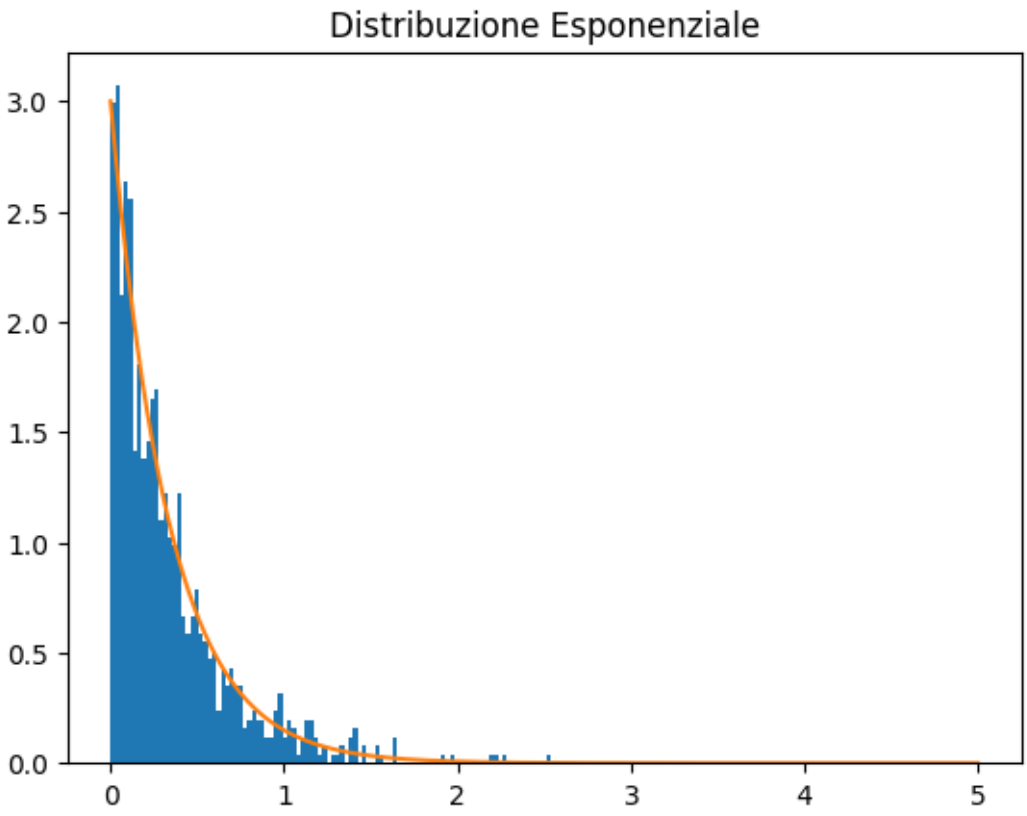
```
assex = np.linspace(a, b, 1000)
assey = uniform.pdf(assex, a, b - a)

plt.hist(Y, density=True, edgecolor="black")
plt.plot(assex, assey)
plt.title("Distribuzione uniforme in [-1,1]")
plt.show()
```



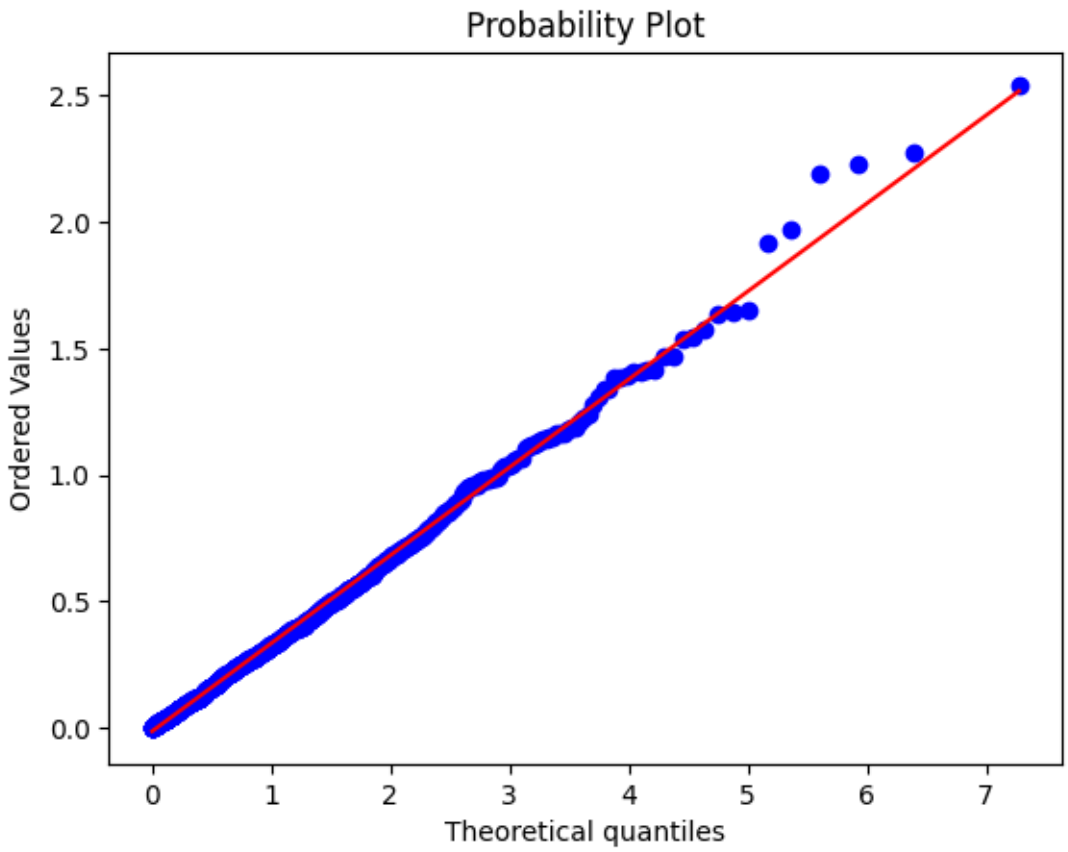
[229]:
```
from scipy.stats import probplot

probplot(Y, dist=uniform, plot=plt)
plt.show()
```



[230]:
```
# Distribuzione esponenziale EXP(3)
from scipy.stats import expon

lam = 3
N = 1000
X = np.random.rand(N)
Y = -np.log(X) / lam

assex = np.linspace(0, 5, 1000)
assey = expon.pdf(assex, 0, 1 / lam)

plt.hist(Y, bins=100, density=True)
plt.plot(assex, assey)
plt.title("Distribuzione Esponenziale")
plt.show()
```

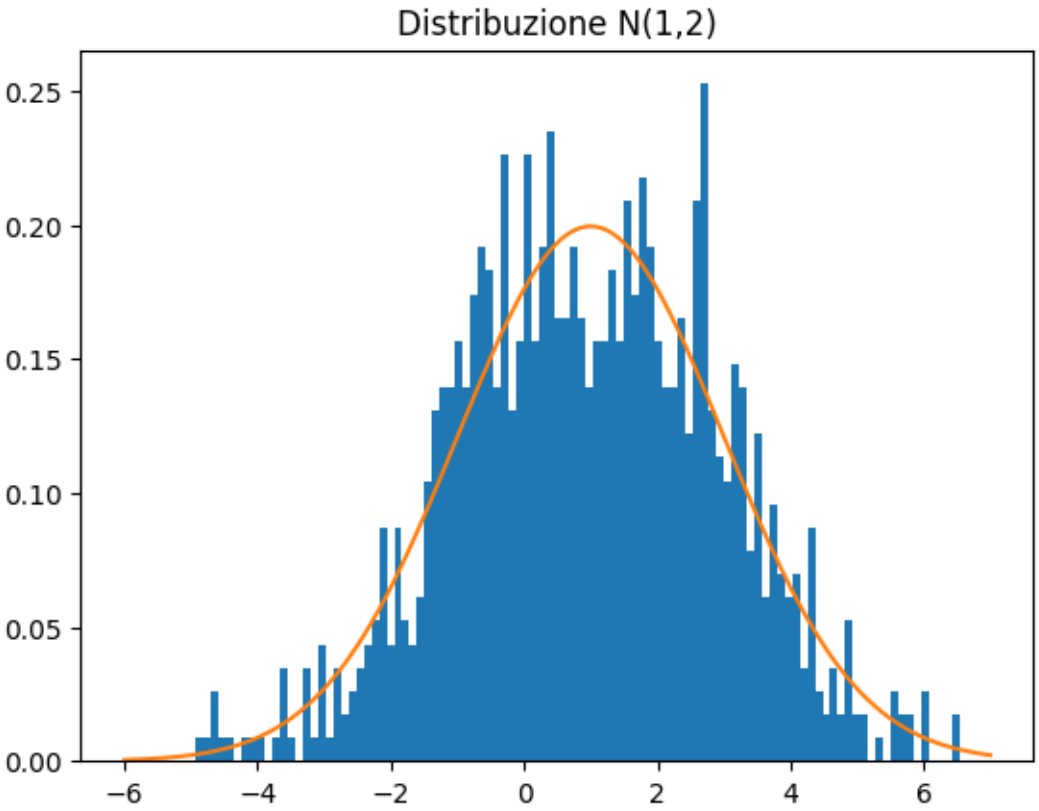


[231]:
```
probplot(Y, dist=expon, plot=plt)
plt.show()
```

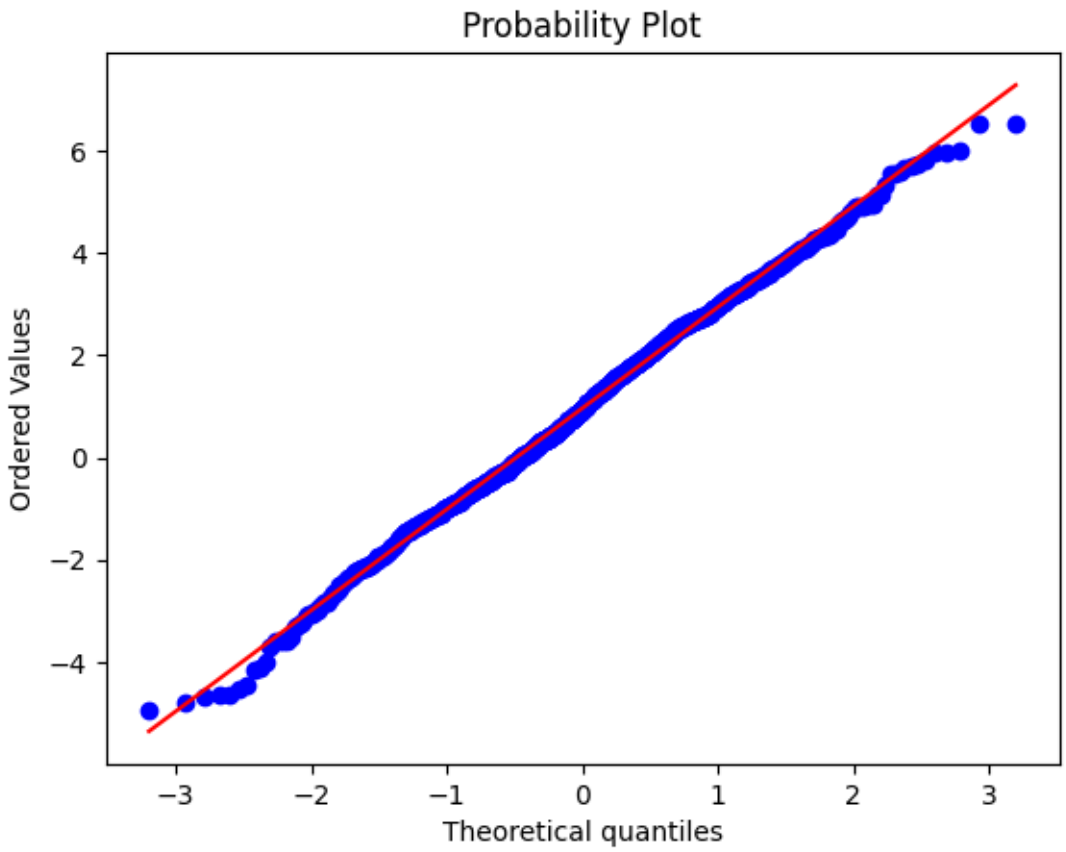


[232]:
```
# Distribuzione Normale N(1,2)
from scipy.stats import norm

mu = 1
sigma = 2
N = 1000

X = np.random.rand(N)

xi1 = X[0 : int(N / 2)]
xi2 = X[int(N / 2) : N]

eta1 = np.sqrt(-2 * np.log(xi1)) * np.cos(2 * np.pi * xi2)
eta2 = np.sqrt(-2 * np.log(xi1)) * np.sin(2 * np.pi * xi2)

Y = np.zeros(N)
Y[0 : int(N / 2)] = mu + sigma * eta1
Y[int(N / 2) : N] = mu + sigma * eta2
```

```
assex = np.linspace(-6, 7, 1000)
assey = norm.pdf(assex, loc=1, scale=2)

plt.hist(Y, bins=100, density=True)
plt.plot(assex, assey)
plt.title("Distribuzione N(1,2)")
plt.show()
```



[233]:
```
probplot(Y, dist=norm, plot=plt)
plt.show()
```



3 Esercizio 3

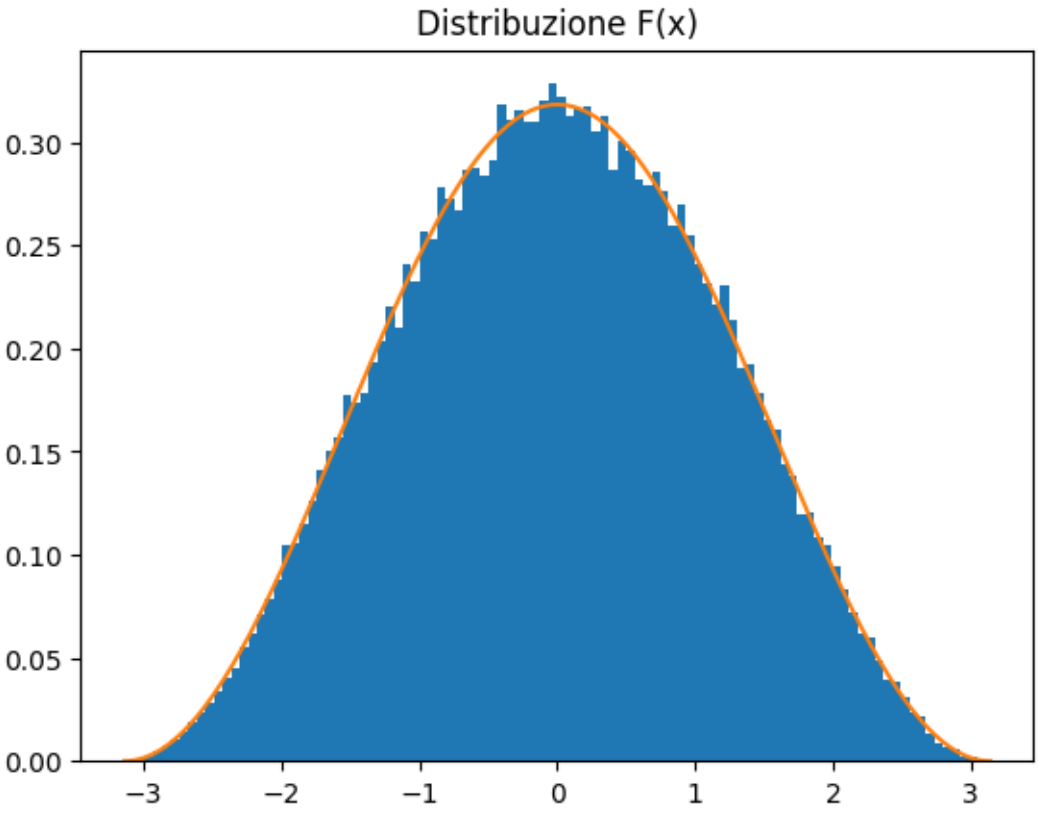
Generare numeri pseudo-casuali con distribuzione

f(x) = 1 + cos(x)/2 con x [− ,].

Eseguire un confronto statistico tra i numeri generati e la distribuzione teorica.

[234]:
```
import numpy as np


def fun(x):
    y = (1 + np.cos(x)) / (2 * np.pi)
    return y


def rigetto(a, b, M):
    while True:
        r1 = np.random.rand()
        r2 = np.random.rand()
```

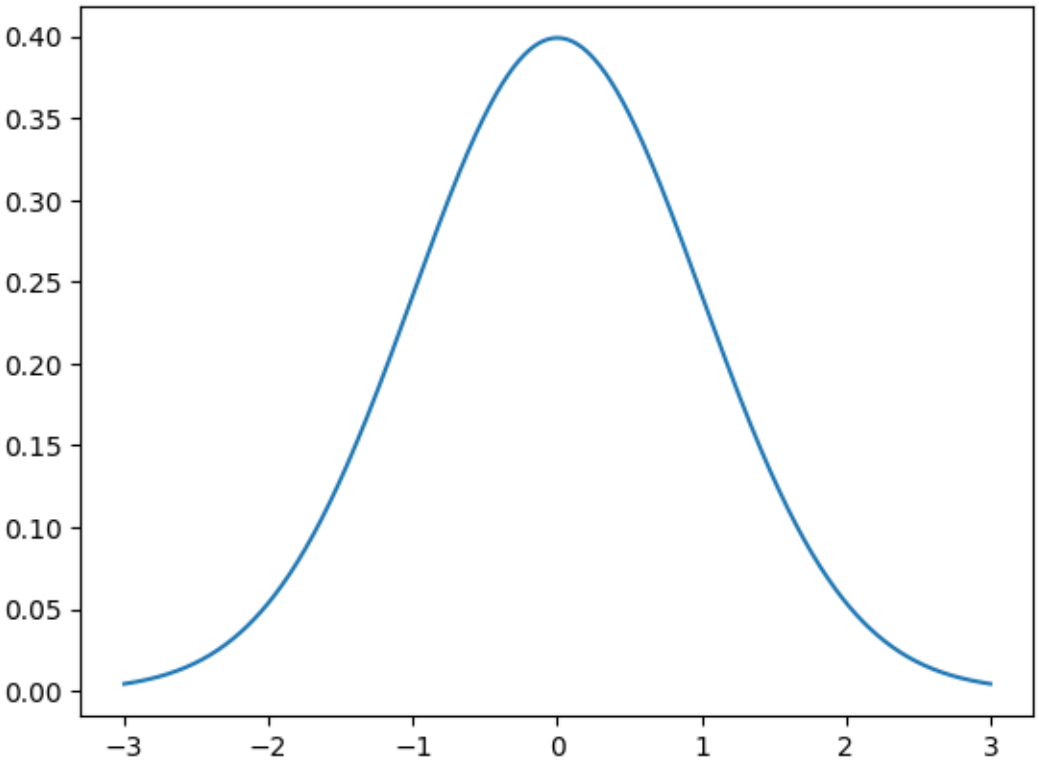
```
        xi = a + r1 * (b - a)
        eta = M * r2
        if eta <= fun(xi):
            break
    return xi


a = -np.pi
b = np.pi
M = 1 / np.pi

N = 100000
X = np.zeros(N)
for i in range(N):
    X[i] = rigetto(a, b, M)

assex = np.linspace(a, b, 1000)
assey = fun(assex)

plt.hist(X, bins=100, density=True)
plt.plot(assex, assey)
plt.title("Distribuzione F(x)")
plt.show()
```



4 Esercizio 4

Sia X N(0, 1). Si calcoli numericamente con il metodo hit or miss la probabilità p = P(0.5 x 2).

[235]:
```
import numpy as np


def fun(x):
    y = np.exp(-(x**2) / 2) / np.sqrt(2 * np.pi)
    return y


a, b = 0.5, 2

assex = np.linspace(-3, 3, 1000)
assey = norm.pdf(assex)
plt.plot(assex, assey)
plt.show()
```



[236]:
```
M = 0.45
N = 100000
NS = 0

for i in range(N):
    r1 = np.random.rand()
    r2 = np.random.rand()
    xi = a + r1 * (b - a)
    eta = r2 * M
    if fun(xi) > eta:
        NS = NS + 1

p = NS / N
I = p * M * (b - a)

phi = norm.cdf(b) - norm.cdf(a)
```

```
0.2857874067778077
```

Esercizi

March 27, 2024

1 Esercizio 1

Data la matrice di transizione

0 1/4 3/4

0 1/2 1/2

3/4 0 1/4

dimostrare che ha un'unica distribuzione stazionaria e determinarla sia analiticamente che con il metodo Monte Carlo confrontando infine i risultati.

[44]:
```
import numpy as np

# Metodo Analitico

P = np.array([[0, 1 / 4, 3 / 4], [0, 1 / 2, 1 / 2], [3 / 4, 0, 1 / 4]])
P2 = np.dot(P, P)

lam, V = np.linalg.eig(P.T)
v = V[:, 1] / np.sum(V[:, 1])

print(v)
```

```
[0.35294118 0.17647059 0.47058824]
```

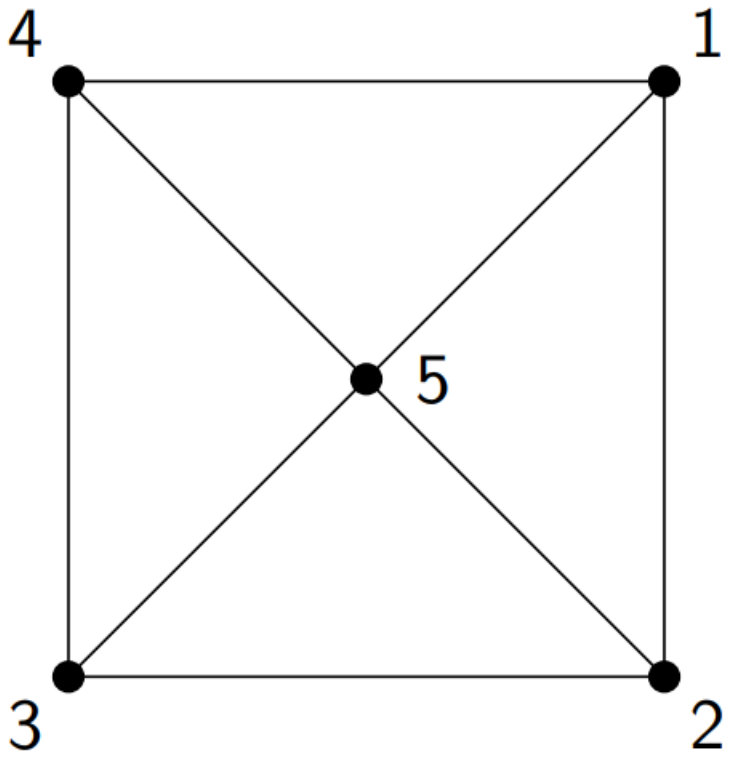
[45]:
```
# Metodo Monte Carlo

n = 3
F = np.zeros(n)
j = np.random.randint(n)
F[j] = 1

N = 100000

for i in range(N):
    j_multi = np.random.multinomial(1, P[j, :])
    j = np.nonzero(j_multi)[0][0]
    F[j] = F[j] + 1
```

```
vv = F / N

print(vv)
```

```
[0.35376 0.17395 0.4723 ]
```

2 Esercizio 2

Consideriamo la passeggiata aleatoria sul grafo della figura seguente.



Determinare la matrice di transizione della catena di Markov associata, dimostrare che la catena è regolare e determinare la distribuzione invariante sia analiticamente che con il metodo Monte Carlo.

[46]:
```
import numpy as np

# Calcolo della matrice di transizione

P = np.array(
    [
        [0, 1 / 3, 0, 1 / 3, 1 / 3],
        [1 / 3, 0, 1 / 3, 0, 1 / 3],
        [0, 1 / 3, 0, 1 / 3, 1 / 3],
        [1 / 3, 0, 1 / 3, 0, 1 / 3],
```

```
        [1 / 4, 1 / 4, 1 / 4, 1 / 4, 0],
    ]
)

print(P)
```

```
[[0.         0.33333333 0.         0.33333333 0.33333333]
 [0.33333333 0.         0.33333333 0.         0.33333333]
 [0.         0.33333333 0.         0.33333333 0.33333333]
 [0.33333333 0.         0.33333333 0.         0.33333333]
 [0.25       0.25       0.25       0.25       0.        ]]
```

[47]:
```
# Metodo analitico

lam, V = np.linalg.eig(P.T)
v = np.real(V[:, 0]) / np.sum(np.real(V[:, 0]))

print(v)
```

```
[0.1875 0.1875 0.1875 0.1875 0.25  ]
```

[48]:
```
# Dimostriamo che la catena è regolare
print(np.dot(P, P))
# Tutti gli elementi sono positivi, quindi la definizione è soddisfatta.
```

```
[[0.30555556 0.08333333 0.30555556 0.08333333 0.22222222]
 [0.08333333 0.30555556 0.08333333 0.30555556 0.22222222]
 [0.30555556 0.08333333 0.30555556 0.08333333 0.22222222]
 [0.08333333 0.30555556 0.08333333 0.30555556 0.22222222]
 [0.16666667 0.16666667 0.16666667 0.16666667 0.33333333]]
```

[49]:
```
n = 5
F = np.zeros(n)
j = np.random.randint(n)
F[j] = 1
N = 100000

for i in range(N):
    j_multi = np.random.multinomial(1, P[j, :])
    j = np.nonzero(j_multi)[0][0]
    F[j] = F[j] + 1

vv = F / N

print(v)
```

```
[0.1875 0.1875 0.1875 0.1875 0.25  ]
```

Esame

March 27, 2024

1 Esercizio 1

Il tempo di vita di un dispositivo meccanico sottoposto a vibrazioni durante un test segue una distribuzione esponenziale con media 400 ore.

1. Qual è la probabilità che il dispositivo fallisca il test in meno di 100 ore?
2. Qual è la probabilità che il dispositivo operi per più di 500 ore prima di rompersi?
3. Sapendo che il dispositivo ha operato per 400 ore senza fallire il test, qual è la probabilità che fallisca nelle prossime 100 ore?
4. Quante ore di funzionamento sono necessarie per affermare che il dispositivo fallisca il test con probabilità superiore al 95%?

[51]:
```
# Punto 1
from scipy.stats import expon

media = 400
punto1 = expon.cdf(100, 0, media)
print(punto1)
```

```
0.22119921692859515
```

[52]:
```
# Punto 2
from scipy.stats import expon

punto2 = 1 - expon.cdf(500, 0, media)
print(punto2)
```

```
0.28650479686019015
```

[53]:
```
# Punto 3
from scipy.stats import expon

punto3 = 1 - expon.cdf(300, 0, media)
print(punto3)
```

```
0.4723665527410147
```

[54]:
```
# Punto 4
from scipy.stats import expon

```

```
media = 400

print(
    "Le ore di funzionamento che sono necessarie per affermare che il␣
 ↪dispositivo fallisca il test con probabilità superiore al 95% sono :",
    expon.ppf(0.95, 0, media),
)
```

```
Le ore di funzionamento che sono necessarie per affermare che il dispositivo
fallisca il test con probabilità superiore al 95% sono : 1198.292909421596
```

Ecco il testo correttamente scritto:

2 Esercizio 2

Si ritiene che i grammi di solidi rimossi da un materiale (y) siano correlati al tempo di asciugatura (x) espresso in ore. Da uno studio sperimentale si ottengono le 10 misurazioni riportate nella seguente tabella.

x	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	6.5	7.0
y	4.3	1.5	1.8	4.9	4.2	4.8	5.8	6.2	7.0	7.9

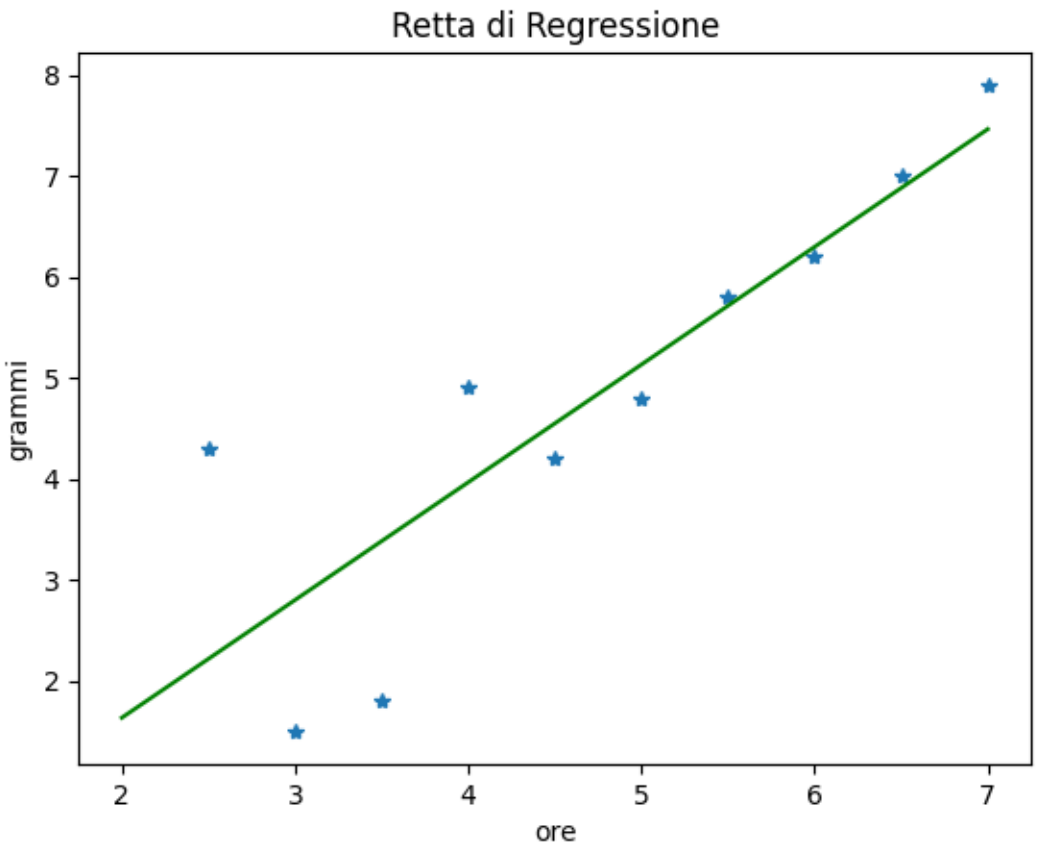
1. Si determinino i coefficienti della retta di regressione e i loro intervalli di confidenza al 95%. Si calcoli il coefficiente di determinazione.
2. Si rappresentino i dati e la retta di regressione in uno stesso grafico.
3. Si stimi la quantità in grammi di solidi rimossi a 4.25 ore.
4. Si effettui il test di indipendenza con un livello di significatività ($= 0.05$) commentandone l'esito.

[55]:
```
# Punto 1
import numpy as np

X = np.array([2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0])
Y = np.array([4.3, 1.5, 1.8, 4.9, 4.2, 4.8, 5.8, 6.2, 7.0, 7.9])

x_bar = np.mean(X)
y_bar = np.mean(Y)
n = X.size

sig_bar_xy = np.sum((X - x_bar) * (Y - y_bar)) / n
sig_bar_x2 = np.sum((X - x_bar) ** 2) / n

b0 = y_bar - sig_bar_xy / sig_bar_x2 * x_bar
b1 = sig_bar_xy / sig_bar_x2

# valori stimati
y_hat = b0 + b1 * X
```

```
r = Y - y_hat
s2 = np.sum(r**2) / (n - 2)

# Intervalli di confidenza al 95%
from scipy.stats import t

alpha = 0.05
s = np.sqrt(s2)
T = t.ppf(1 - alpha / 2, (n - 2))

sig_x = np.sqrt(sig_bar_x2)

b_0_sx = b0 - s * np.sqrt(1 / n + x_bar**2 / (n * sig_bar_x2)) * T
b_0_dx = b0 + s * np.sqrt(1 / n + x_bar**2 / (n * sig_bar_x2)) * T

b_1_sx = b1 - s / (sig_x * np.sqrt(n)) * T
b_1_dx = b1 + s / (sig_x * np.sqrt(n)) * T


print("Intervallo di confidenza di b_0: [{},{}]".format(b_0_sx, b_0_dx))
print("Intervallo di confidenza di b_1: [{},{}]".format(b_1_sx, b_1_dx))

# Coefficente di determinazione
sig_y_2 = np.sum((Y - y_bar) ** 2) / n
R2 = sig_bar_xy**2 / (sig_bar_x2 * sig_y_2)
print("Coefficiente di determinazione: ", R2)
```

```
Intervallo di confidenza di b_0: [-3.4961586033066223,2.098582845730862]
Intervallo di confidenza di b_1: [0.6023426015180314,1.729778610603181]
Coefficiente di determinazione:  0.7398627473553604
```

[56]:
```
# Punto 2

import matplotlib.pyplot as plt

min = np.round(np.min(X))
max = np.round(np.max(X))
assex = np.linspace(min, max, 1000)
assey = b0 + b1 * assex

plt.plot(X, Y, "*")
plt.plot(assex, assey, color="green")
plt.xlabel("ore")
plt.ylabel("grammi")
plt.title("Retta di Regressione")
plt.show()
```



[57]:
```
# Punto 3

punto3 = b0 + b1 * 4.25
print(punto3)
```

```
4.256969696969697
```

[58]:
```
# Punto 4
from scipy.stats import t

alpha = 0.05

t = t.ppf(1 - alpha / 2, n - 2)
T = np.abs(np.sqrt(n) * b1 / s * sig_x)
print("T0 : ", T)
print("t:", t)
print("Siccome T0>t possiamo rigettare l'ipotesi")
```

```
T0 :  4.770010107193931
t: 2.3060041350333704
Siccome T0>t possiamo rigettare l'ipotesi
```

3 Esercizio 3

Una variabile aleatoria X ha densità:

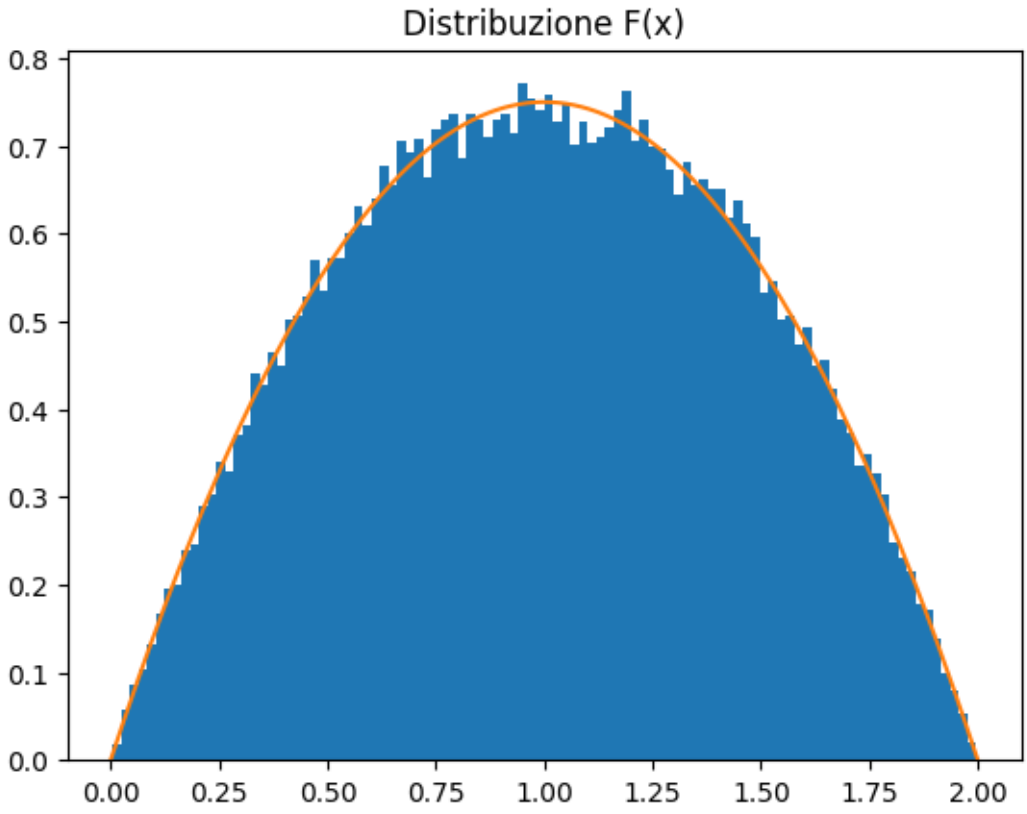
$$f(x) = \begin{cases} -\frac{3}{4}x^2 + \frac{3}{2}x & x \in [0, 2] \\ 0 & \text{altrimenti} \end{cases}$$

1. Descrivere e implementare un algoritmo per generare numeri pseudo-casuali con distribuzione X utilizzando il metodo del rigetto.
2. Generare numericamente 10^5 di questi numeri e costruire un istogramma. Verificare infine che essi seguano approssimativamente la distribuzione assegnata sovrapponendo all'istogramma il grafico di f(x) in [0, 2].

[59]:
```
import numpy as np


def fun(x):
    return (-3 / 4) * (x**2) + 3 / 2 * x


def rigetto(a, b, M):
    while True:
        r1 = np.random.rand()
        r2 = np.random.rand()
        xi = a + r1 * (b - a)
        eta = M * r2
        if eta <= fun(xi):
            break
    return xi


a = 0
b = 2
M = fun(1)

N = 100000
X = np.zeros(N)
for i in range(N):
    X[i] = rigetto(a, b, M)
```

[60]:
```
assex = np.linspace(a, b, 1000)
assey = fun(assex)
```

```
plt.hist(X, bins=100, density=True)
plt.plot(assex, assey)
plt.title("Distribuzione F(x)")
plt.show()
```



Esame

March 27, 2024

1 Esercizio 1

È noto che il numero di pezzi guasti fabbricati in una giornata di lavoro di una catena di produzione A segue una distribuzione di Poisson di media 2.

1. Qual è la probabilità che in un giorno siano stati prodotti esattamente 3 pezzi guasti?
2. Qual è la probabilità che in un giorno siano stati prodotti tra 2 e 5 pezzi guasti (estremi inclusi)?

Si mette in opera una nuova catena di produzione B. È noto che il numero di pezzi guasti fabbricati in una giornata di lavoro mediante B segue una distribuzione di Poisson di media 1.5.

3. Si trovi la legge della variabile aleatoria che conta complessivamente il numero di pezzi guasti prodotti (cioè provenienti indifferentemente da A o da B) e si calcoli la sua media e la sua varianza.
4. Qual è la probabilità che in un giorno siano stati prodotti complessivamente un numero di pezzi guasti compreso tra 3 e 6 (estremi inclusi)?

[2]:
```
import numpy as np
from scipy.stats import poisson

lam = 2

# Punto 1

punto1 = poisson.pmf(3, lam)
print("Probabilità di trovare 3 pezzi guasti: ", punto1)

# Punto 2

punto2 = (
    poisson.pmf(2, lam)
    + poisson.pmf(3, lam)
    + poisson.pmf(4, lam)
    + poisson.pmf(5, lam)
)

"""
for i in range(2,6):
```

```
    punto2 += poisson.pmf(i,lam)
"""

print(punto2)

# Punto 3

mu = 1.5

# lam>0 e mu>0 -> X+Y Poisson(lam+mu)
# quindi la nuova v.a. sarà Z Poisson (4.5)
# la cui media e varianza sarà uguale a 4.5

# Punto 4

punto4 = 0
"""
for i in range(3, 7):
    punto4 += poisson.pmf(i, lam + mu)
print(punto4)
"""

punto4 = 0
punto4 = (
    poisson.pmf(3, lam + mu)
    + poisson.pmf(4, lam + mu)
    + poisson.pmf(5, lam + mu)
    + poisson.pmf(6, lam + mu)
)
print(punto4)
```

```
Probabilità di trovare 3 pezzi guasti:  0.18044704431548356
0.5774305418095474
0.6138647041089124
```

2 Esercizio 2

Il contenuto di sodio (in milligrammi) di 30 scatole di cereali è riportato di seguito:

131.15, 130.69, 130.91, 129.54, 129.64, 128.77, 130.72, 128.33, 128.24, 129.65, 130.14, 129.29, 128.71, 129.00, 129.39, 130.42, 129.53, 130.12, 129.78, 130.92, 131.15, 130.69, 130.91, 129.54, 129.64, 128.77, 130.72, 128.33, 128.24, 129.65.

1. Si calcoli la media campionaria, la deviazione standard e l'intervallo di confidenza per la media con livello di fiducia 0.01.
2. Rappresentare graficamente i dati mediante un istogramma e mediante un box-plot.
3. Si testi l'ipotesi che il contenuto medio di sodio sia di 130 mg utilizzando ($= 0.05$). Si calcoli il p-value del test precedente.

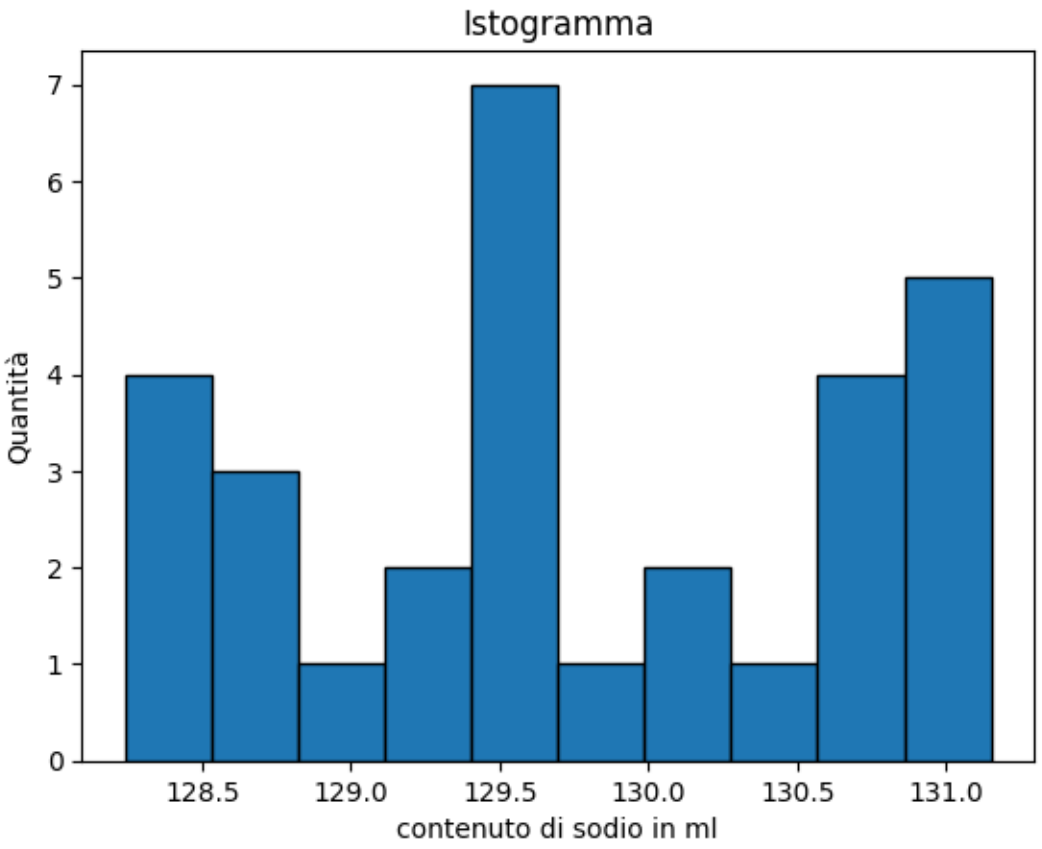
4. È possibile affermare che il contenuto di sodio è distribuito normalmente nelle scatole? Giustificare la risposta.

[3]:
```
import numpy as np
from scipy.stats import norm

A = np.array(
    [
        131.15,
        130.69,
        130.91,
        129.54,
        129.64,
        128.77,
        130.72,
        128.33,
        128.24,
        129.65,
        130.14,
        129.29,
        128.71,
        129.00,
        129.39,
        130.42,
        129.53,
        130.12,
        129.78,
        130.92,
        131.15,
        130.69,
        130.91,
        129.54,
        129.64,
        128.77,
        130.72,
        128.33,
        128.24,
        129.65,
    ]
)
n = A.size
# Punto 1
media = np.mean(A)
deviazione_standard = np.std(A, ddof=1)
alpha = 0.01
PHI = norm.ppf(1 - alpha / 2)
```

```
il = media - ((deviazione_standard / np.sqrt(n)) * PHI)
ir = media + ((deviazione_standard / np.sqrt(n)) * PHI)
print(il)
print(ir)
```

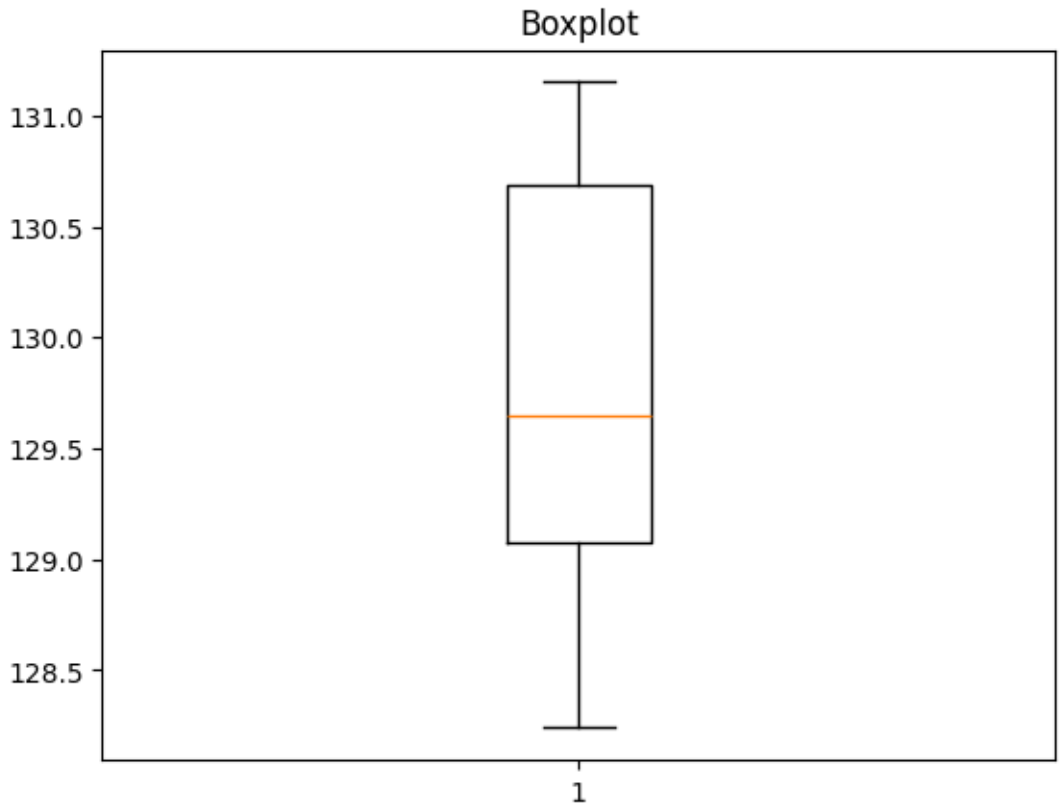
```
129.31560733504844
130.18972599828484
```

[4]:
```
# Punto 2
import matplotlib.pyplot as plt

plt.hist(A, bins=10, edgecolor="black")
plt.xlabel("contenuto di sodio in ml")
plt.ylabel("Quantità")
plt.title("Istogramma")
plt.show()
```



[5]:
```
import matplotlib.pyplot as plt

plt.boxplot(A)
```

```
plt.title("Boxplot")
plt.show()
```



[6]:
```
from scipy.stats import t

# Punto 3
alpha = 0.05
mu = 130
T = ((media - mu) / deviazione_standard) * np.sqrt(n)
t = t.ppf(1 - alpha / 2, n - 1)
print("T0: ", np.abs(T))
print("t: ", t)

print("T0 < t: Non si può rigettare l'ipotesi")
```

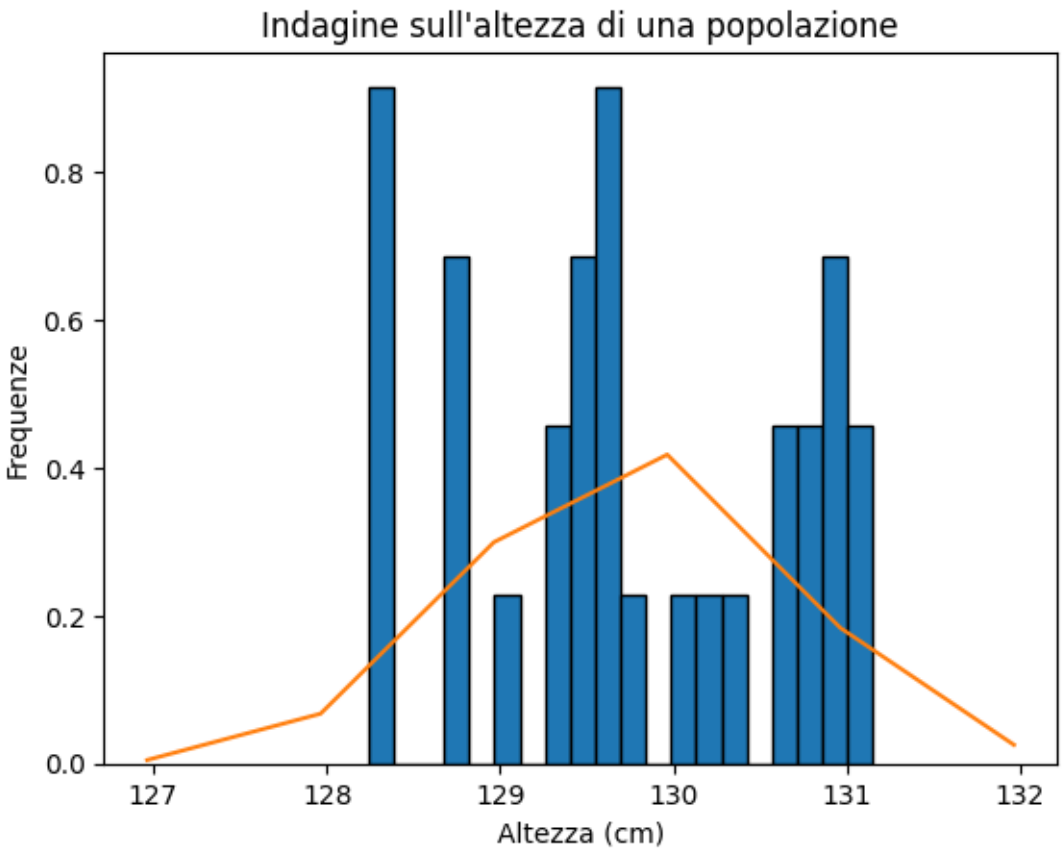
```
T0:  1.4576703931379382
t:  2.045229642132703
T0 < t: Non si può rigettare l'ipotesi
```

[7]:
```
from scipy.stats import t
```

```
# p-value
p_value = 2 * (1 - t.cdf(T, n - 1))
print("p-value: ", p_value)
```

```
p-value:  1.844323633977642
```

[8]:
```
# Punto 4
from scipy.stats import norm

assex = np.arange(media - (3 * deviazione_standard), media + (3 *␣
 ↪deviazione_standard))
assey = norm.pdf(assex, media, deviazione_standard)

plt.hist(A, 20, edgecolor="black", density="True")
plt.plot(assex, assey)

plt.title("Indagine sull'altezza di una popolazione")
plt.xlabel("Altezza (cm)")
plt.ylabel("Frequenze")

plt.show()

# Non segue una normale
```



3 Esercizio 3

Si consideri la catena di Markov sui vertici di un triangolo equilatero definita come segue. Ad ogni istante ci si può spostare da un vertice a quello adiacente in senso orario con probabilità (p) e in senso antiorario con probabilità 1 - p , con $0 < p < 1$.

1. Determinare la matrice di transizione.
2. Fissato p=1/3, mostrare che la catena è regolare.
3. Fissato p=1/3, determinare la distribuzione stazionaria.
4. Indicato con E = {1, 2, 3} l'insieme degli stati, determinare il valore di p per cui la distribuzione uniforme su E sia reversibile.

[9]:
```
# Punto 1 Questa è la matrice di transizione
# [0 p 1-p]
# [1-p 0 p]
# [p 1-p 0]
```

[10]:
```
import numpy as np
M = np.array([[0,1/3,2/3],[2/3,0,1/3],[1/3,2/3,0]])
print(np.dot(M,M))
#Sono tutti positivi quindi è regolare
```

```
[[0.44444444 0.44444444 0.11111111]
 [0.11111111 0.44444444 0.44444444]
 [0.44444444 0.11111111 0.44444444]]
```

[11]:
```
#Punto 3
#primo metodo
lam, V = np.linalg.eig(M.T)
v = np.real(V[:, 0]) / np.sum(np.real(V[:, 0]))
print(v)
```

```
[0.33333333 0.33333333 0.33333333]
```

[12]:
```
#secondo metodo
import numpy as np
n=3
F=np.zeros(n)
j=np.random.randint(n)
F[j]=1
N=100000
for i in range(N):
        j_multi=np.random.multinomial(1,M[j,:])
        j=np.nonzero(j_multi)[0][0]
        F[j]=F[j]+1
```

```
vv=F/N
print(vv)
```

```
[0.33313 0.33375 0.33313]
```

[13]:
```
#Punto 4
import numpy as np

# Definisci la tua matrice di transizione in funzione di p
def M(p):
    return np.array([[0, p, 1-p], [1-p, 0, p], [p, 1-p, 0]])

# Per una catena di Markov con un insieme di stati E = {1, 2, 3},
# la distribuzione uniforme su E è  = (1/3, 1/3, 1/3).
# La distribuzione  è reversibile per la catena di Markov se e solo se
#  i Pij =  j Pji per ogni coppia di stati i, j in E.
# Sostituendo  e P in questa equazione otteniamo 1/3 Pij = 1/3 Pji,
# da cui segue che Pij = Pji per ogni coppia di stati i, j in E.
# Quindi, la distribuzione uniforme su E è reversibile per la catena di Markov
# se e solo se la matrice di transizione P è simmetrica.

# Dato che la matrice di transizione P è definita in termini di p,
# possiamo risolvere l'equazione Pij = Pji per p per trovare il valore di p
# per cui la matrice di transizione P è simmetrica.
# In particolare, abbiamo p = P12 = P21 = 1 - P13 = 1 - P31 = P23 = P32 = 1 - p.

# Risolvendo queste equazioni per p otteniamo p = 1/2.
# Quindi, la distribuzione uniforme su E è reversibile per la catena di Markov
# se e solo se p = 1/2.

# Ricorda che questa soluzione è valida solo se 0 < p < 1,
# come specificato nel problema.
# Se p = 1/2 non soddisfa questa condizione,
# allora non esiste un valore di p per cui la distribuzione uniforme su E
# sia reversibile per la catena di Markov.
```