Introduzione

Il codice del progetto è open source con licenza MIT ed è possibile trovarlo al seguente link: https://github.com/OB-UNISA/Statistica. E' possibile visualizzare una versione interattiva online al seguente link: https://datalore.jetbrains.com/view/notebook/JY0gPFblPn0G4xonfjdSWh.

I dati unidimensionali esaminati sono quelli del peso corporeo nel periodo di tre anni di un uomo di età 24 attuali.
I dati bidimensionali riguardano il peso corporeo della medesima persona nello stesso periodo e le calorie assunte giornalmente.

Il peso corporeo è stato registrato con due bilance differenti, una per il primo anno e un'altra nei restanti e quindi vi potrebbe essere una differenza di misura.
I dati dei primi mesi sulle calorie assunte non sono molto precisi in quanto la quantità del cibo non veniva pesata, ma approssimata.

Allo stato dell'arte, importando i dati da Samsung Health, è possibile visualizzare le proprie statistiche. Questo potrebbe non funzionare in futuro se la forma dei dati venisse cambiata, ossia il parsing dei dati non funzionerebbe più. I commenti sui dati, tuttavia, risulteranno inefficienti in quanto non sono dinamici, bensì basati su un campione di esso.

Poiché i dati sono molti riguardanti il peso corporeo, circa 500, e sulle calorie assunte, quasi 1000, ne verrà preso un campione calcolato nel seguente modo:

- Per il peso corporeo, il campione sarà il peso medio delle pesate in ogni settimana.
  Quindi x_i = peso medio nella settimana i.
- Per le calorie assunte, il campione sarà la media delle calorie assunte in ogni settimana.
  Quindi y_i = media delle calorie assunte nella settimana i.

Nota: Se nella settimana i non vi sono dati sul peso, essa verrà saltata e quindi anche le rispettive calorie assunte.

Si è deciso di usare la media campionaria e non la mediana perché i picchi di valore sono importanti da considerare.

Installazione dipendenze

In [115…
```
!pip install matplotlib
!pip install pandas
```

```
Requirement already satisfied: matplotlib in c:\python397\lib\site-packages (3.5.2)
Requirement already satisfied: numpy>=1.17 in c:\python397\lib\site-packages (from matplotlib) (1.22.4)
Requirement already satisfied: cycler>=0.10 in c:\python397\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\python397\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\python397\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: packaging>=20.0 in c:\python397\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\python397\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in c:\python397\lib\site-packages (from matplotlib) (4.33.3)
Requirement already satisfied: pillow>=6.2.0 in c:\python397\lib\site-packages (from matplotlib) (9.1.1)
Requirement already satisfied: six>=1.5 in c:\python397\lib\site-packages (from python-dateutil>=2.7->matplotli
b) (1.16.0)
Requirement already satisfied: pandas in c:\python397\lib\site-packages (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\python397\lib\site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\python397\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.18.5 in c:\python397\lib\site-packages (from pandas) (1.22.4)
Requirement already satisfied: six>=1.5 in c:\python397\lib\site-packages (from python-dateutil>=2.8.1->pandas)
(1.16.0)
```

In [116…
```
import matplotlib.pyplot as plt
import pandas as pd
import csv
from math import sqrt
```

Parsing dei dati del peso corporeo

È stato necessario arrotondare i valori del peso corporeo perché altrimenti si avrebbe avuto il numero di modalità del carattere quasi uguale all'ampiezza del dato.

In [117…
```
with open('peso.csv', 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    peso = []
    reader.__next__()
    reader.__next__()
    for row in reader:
        peso.append([row[6], float(row[4])])

df_peso = pd.DataFrame(peso, columns=['data', 'peso'])
df_peso['data'] = pd.to_datetime(df_peso['data'])
df_peso = df_peso.groupby(pd.Grouper(key='data', freq='W')).mean().round(0)
df_peso = df_peso.reset_index()
df_peso
```

Out[117]:

	data	peso
0	2019-12-29	140.0
1	2020-01-05	NaN
2	2020-01-12	140.0
3	2020-01-19	NaN
4	2020-01-26	139.0
...	...	...
123	2022-05-08	105.0
124	2022-05-15	104.0
125	2022-05-22	104.0
126	2022-05-29	103.0
127	2022-06-05	102.0

128 rows × 2 columns

Sono state eliminate le settimane in cui non vi erano dati.

In [118…
```
x_not_sorted = []
date_valide = []
for i in range(len(df_peso)):
    if not pd.isnull(df_peso.iloc[i]['peso']):
        x_not_sorted.append(int(df_peso.iloc[i]['peso']))
        date_valide.append(df_peso.iloc[i]['data'])
x = sorted(x_not_sorted)
n = len(x)
print(f'Ampiezza del dato: {n}')
for i, x_i in enumerate(x):
    print(f'x_({i + 1})={x_i}', end=' ')
```

```
Ampiezza del dato: 115
x_(1)=102 x_(2)=102 x_(3)=103 x_(4)=103 x_(5)=103 x_(6)=103 x_(7)=103 x_(8)=103 x_(9)=103 x_(10)=103 x_(11)=103
x_(12)=104 x_(13)=104 x_(14)=104 x_(15)=104 x_(16)=104 x_(17)=104 x_(18)=104 x_(19)=104 x_(20)=104 x_(21)=105 x
_(22)=105 x_(23)=105 x_(24)=105 x_(25)=105 x_(26)=105 x_(27)=105 x_(28)=105 x_(29)=105 x_(30)=105 x_(31)=106 x_
(32)=106 x_(33)=106 x_(34)=106 x_(35)=106 x_(36)=107 x_(37)=107 x_(38)=108 x_(39)=108 x_(40)=108 x_(41)=109 x_
(42)=109 x_(43)=110 x_(44)=111 x_(45)=111 x_(46)=112 x_(47)=112 x_(48)=113 x_(49)=114 x_(50)=115 x_(51)=116 x_
(52)=117 x_(53)=117 x_(54)=117 x_(55)=117 x_(56)=117 x_(57)=117 x_(58)=117 x_(59)=118 x_(60)=118 x_(61)=118 x_
(62)=118 x_(63)=118 x_(64)=118 x_(65)=118 x_(66)=119 x_(67)=120 x_(68)=121 x_(69)=121 x_(70)=121 x_(71)=121 x_
(72)=122 x_(73)=122 x_(74)=122 x_(75)=122 x_(76)=122 x_(77)=123 x_(78)=124 x_(79)=124 x_(80)=125 x_(81)=125 x_
(82)=125 x_(83)=126 x_(84)=126 x_(85)=126 x_(86)=126 x_(87)=126 x_(88)=127 x_(89)=127 x_(90)=127 x_(91)=128 x_
(92)=128 x_(93)=129 x_(94)=129 x_(95)=129 x_(96)=130 x_(97)=130 x_(98)=131 x_(99)=131 x_(100)=132 x_(101)=132 x
_(102)=133 x_(103)=133 x_(104)=134 x_(105)=134 x_(106)=135 x_(107)=136 x_(108)=137 x_(109)=137 x_(110)=138 x_(1
11)=138 x_(112)=139 x_(113)=139 x_(114)=140 x_(115)=140
```

Costruzione della tabella delle frequenze

Calcolo delle modalità

In [119…
```
v_x = [x[0]]
for i in range(1, len(x)):
    if v_x[-1] != x[i]:
        v_x.append(x[i])

k_x = len(v_x)

for i, v_i in enumerate(v_x):
    print(f'v_{i + 1}={v_i}', end=' ')
```

```
v_1=102 v_2=103 v_3=104 v_4=105 v_5=106 v_6=107 v_7=108 v_8=109 v_9=110 v_10=111 v_11=112 v_12=113 v_13=114 v_1
4=115 v_15=116 v_16=117 v_17=118 v_18=119 v_19=120 v_20=121 v_21=122 v_22=123 v_23=124 v_24=125 v_25=126 v_26=1
27 v_27=128 v_28=129 v_29=130 v_30=131 v_31=132 v_32=133 v_33=134 v_34=135 v_35=136 v_36=137 v_37=138 v_38=139
v_39=140
```

Calcolo della frequenza assoluta delle modalità

In [120…
```
f_x = [1]
for i in range(1, n):
    if x[i - 1] == x[i]:
        f_x[-1] += 1
    else:
        f_x.append(1)

for i, f_i in enumerate(f_x):
    print(f'f_{i + 1}={f_i}', end=' ')
```

```
f_1=2 f_2=9 f_3=9 f_4=10 f_5=5 f_6=2 f_7=3 f_8=2 f_9=1 f_10=2 f_11=2 f_12=1 f_13=1 f_14=1 f_15=1 f_16=7 f_17=7
f_18=1 f_19=1 f_20=4 f_21=5 f_22=1 f_23=2 f_24=3 f_25=5 f_26=3 f_27=2 f_28=3 f_29=2 f_30=2 f_31=2 f_32=2 f_33=2
f_34=1 f_35=1 f_36=2 f_37=2 f_38=2 f_39=2
```

Calcolo della frequenza cumulativa assoluta delle modalità

In [121…
```
F_x = [f_x[0]]
for i in range(1, k_x):
    # Uso della relazione di ricorrenza
    F_x.append(F_x[-1] + f_x[i])

for i, F_i in enumerate(F_x):
    print(f'F_{i + 1}={F_i}', end=' ')
```

```
F_1=2 F_2=11 F_3=20 F_4=30 F_5=35 F_6=37 F_7=40 F_8=42 F_9=43 F_10=45 F_11=47 F_12=48 F_13=49 F_14=50 F_15=51 F
_16=58 F_17=65 F_18=66 F_19=67 F_20=71 F_21=76 F_22=77 F_23=79 F_24=82 F_25=87 F_26=90 F_27=92 F_28=95 F_29=97
F_30=99 F_31=101 F_32=103 F_33=105 F_34=106 F_35=107 F_36=109 F_37=111 F_38=113 F_39=115
```

Calcolo della frequenza relativa delle modalità

In [122…
```
p_x = []
for f_i in f_x:
    p_x.append(f_i / n)

for i, p_i in enumerate(p_x):
    print(f'p_{i + 1}={p_i:.3f}', end=' ')
```

```
p_1=0.017 p_2=0.078 p_3=0.078 p_4=0.087 p_5=0.043 p_6=0.017 p_7=0.026 p_8=0.017 p_9=0.009 p_10=0.017 p_11=0.017
p_12=0.009 p_13=0.009 p_14=0.009 p_15=0.009 p_16=0.061 p_17=0.061 p_18=0.009 p_19=0.009 p_20=0.035 p_21=0.043 p
_22=0.009 p_23=0.017 p_24=0.026 p_25=0.043 p_26=0.026 p_27=0.017 p_28=0.026 p_29=0.017 p_30=0.017 p_31=0.017 p_
32=0.017 p_33=0.017 p_34=0.009 p_35=0.009 p_36=0.017 p_37=0.017 p_38=0.017 p_39=0.017
```

Calcolo della frequenza cumulativa relativa delle modalità

In [123…
```
P_x = []
for F_i in F_x:
    P_x.append(F_i / n)

for i, P_i in enumerate(P_x):
    print(f'P_{i + 1}={P_i:.3f}', end=' ')
```

```
P_1=0.017 P_2=0.096 P_3=0.174 P_4=0.261 P_5=0.304 P_6=0.322 P_7=0.348 P_8=0.365 P_9=0.374 P_10=0.391 P_11=0.409
P_12=0.417 P_13=0.426 P_14=0.435 P_15=0.443 P_16=0.504 P_17=0.565 P_18=0.574 P_19=0.583 P_20=0.617 P_21=0.661 P
_22=0.670 P_23=0.687 P_24=0.713 P_25=0.757 P_26=0.783 P_27=0.800 P_28=0.826 P_29=0.843 P_30=0.861 P_31=0.878 P_
32=0.896 P_33=0.913 P_34=0.922 P_35=0.930 P_36=0.948 P_37=0.965 P_38=0.983 P_39=1.000
```

Tabella delle frequenze

In [124…
```
data_table = []
for i in range(k_x):
    data_table.append([i + 1, v_x[i], f_x[i], p_x[i], F_x[i], P_x[i]])

df_data_table = pd.DataFrame(data_table, columns=['i', 'v_i', 'f_i', 'p_i', 'F_i', 'P_i'])
df_data_table.set_index('i', inplace=True)
df_data_table
```

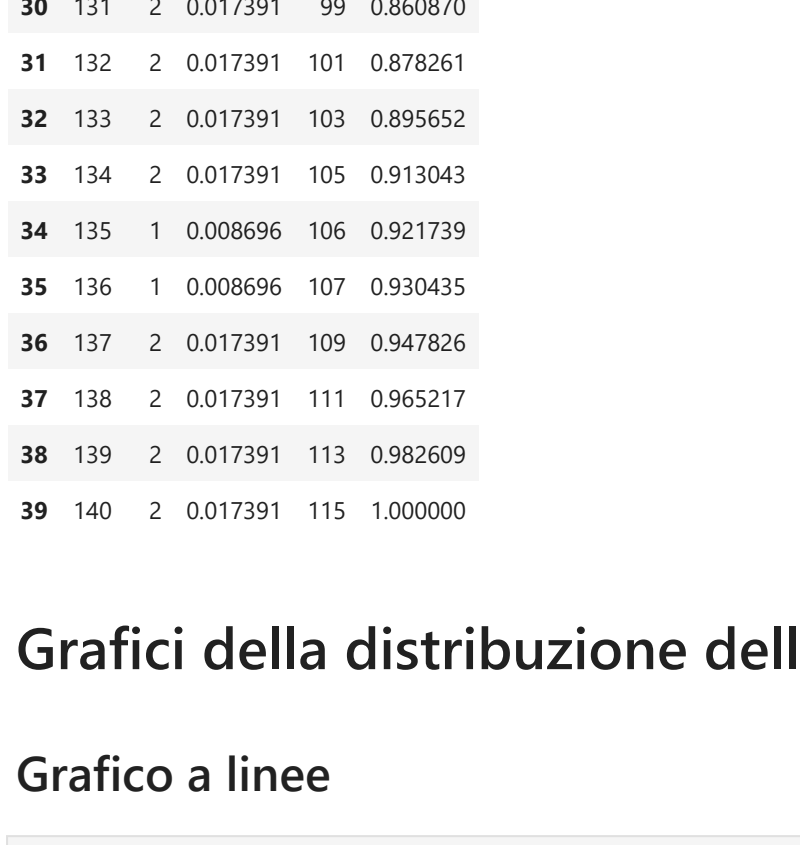
Out[124]:

i	v_i	f_i	p_i	F_i	P_i
1	102	2	0.017391	2	0.017391
2	103	9	0.078261	11	0.095652
3	104	9	0.078261	20	0.173913
4	105	10	0.086957	30	0.260870
5	106	5	0.043478	35	0.304348
6	107	2	0.017391	37	0.321739
7	108	3	0.026087	40	0.347826
8	109	2	0.017391	42	0.365217
9	110	1	0.008696	43	0.373913
10	111	2	0.017391	45	0.391304
11	112	2	0.017391	47	0.408696
12	113	1	0.008696	48	0.417391
13	114	1	0.008696	49	0.426087
14	115	1	0.008696	50	0.434783
15	116	1	0.008696	51	0.443478
16	117	7	0.060870	58	0.504348
17	118	7	0.060870	65	0.565217
18	119	1	0.008696	66	0.573913
19	120	1	0.008696	67	0.582609
20	121	4	0.034783	71	0.617391
21	122	5	0.043478	76	0.660870
22	123	1	0.008696	77	0.669565
23	124	2	0.017391	79	0.686957
24	125	3	0.026087	82	0.713043
25	126	5	0.043478	87	0.756522
26	127	3	0.026087	90	0.782609
27	128	2	0.017391	92	0.800000
28	129	3	0.026087	95	0.826087
29	130	2	0.017391	97	0.843478
30	131	2	0.017391	99	0.860870
31	132	2	0.017391	101	0.878261
32	133	2	0.017391	103	0.895652
33	134	2	0.017391	105	0.913043
34	135	1	0.008696	106	0.921739
35	136	1	0.008696	107	0.930435
36	137	2	0.017391	109	0.947826
37	138	2	0.017391	111	0.965217
38	139	2	0.017391	113	0.982609
39	140	2	0.017391	115	1.000000

Grafici della distribuzione delle frequenze assolute

Grafico a linee

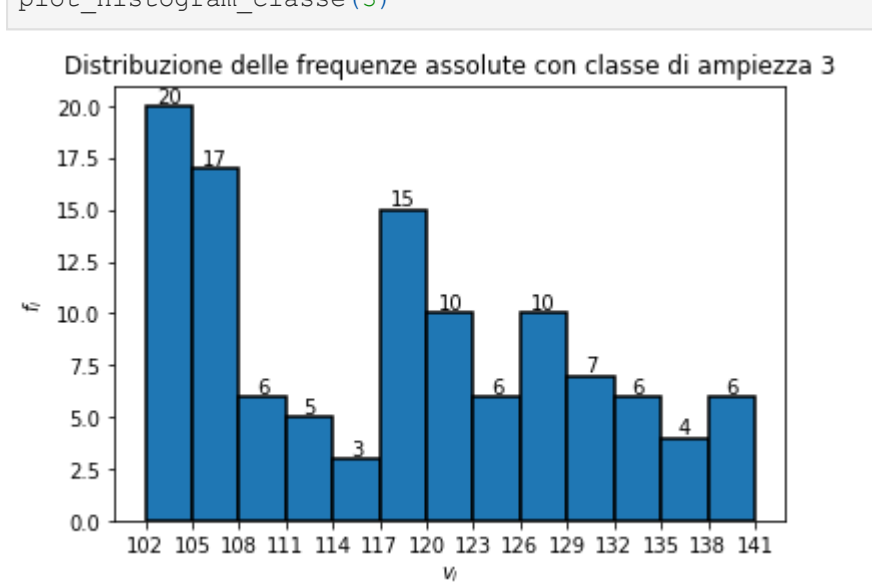
In [125…
```
fig, ax = plt.subplots()
ax.plot(v_x, f_x, '-o')
ax.set_title('Distribuzione delle frequenze assolute')
ax.set_xlabel(r'$v_i$')
ax.set_ylabel(r'$f_i$')
ax.grid()

plt.show()
```



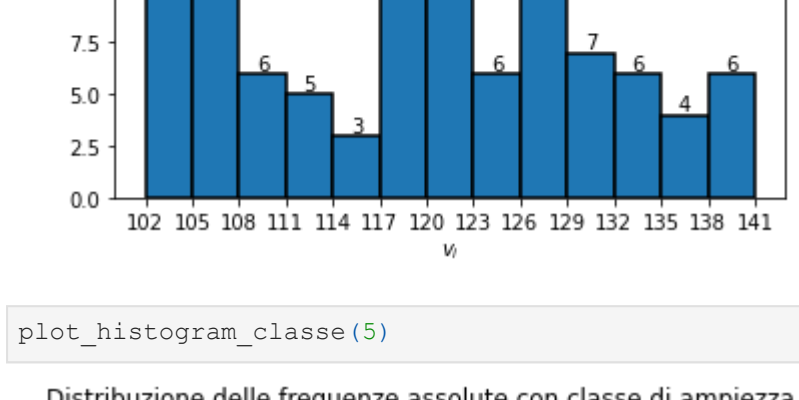
Istogramma

In [126…
```
def plot_histogram_classe(ampiezza_classe):
    min_value = x[0]
    max_value = x[n - 1]

    fig, ax = plt.subplots()
    values, bins, bars = ax.hist(x, bins=list(range(min_value, max_value + ampiezza_classe, ampiezza_classe)), 

    ax.set_xticks(bins)
    ax.bar_label(bars)
    ax.set_title(f'Distribuzione delle frequenze assolute con classe di ampiezza {ampiezza_classe}')
    ax.set_xlabel(r'$v_i$')
    ax.set_ylabel(r'$f_i$')

    plt.show()
```

In [127…
```
plot_histogram_classe(3)
```



In [128…
```
plot_histogram_classe(5)
```



Indici di posizione

Calcolo della media campionaria

In [129…
```
x_mean = 0
for x_i in x:
    x_mean += x_i
x_mean = x_mean / n

print(f'Media campionaria: {x_mean}')
```

```
Media campionaria: 117.3391304347826
```

Calcolo della media pesata

In [130…
```
x_mean_pesata = 0
for i in range(k_x):
    x_mean_pesata += (f_x[i] / n) * v_x[i]

print(f'Media pesata: {x_mean_pesata}')
```

```
Media pesata: 117.33913043478262
```

Calcolo della mediana campionaria

In [131…
```
if n % 2 == 0:
    x_mediana = (x[n // 2] + x[n // 2 - 1]) / 2
else:
    x_mediana = x[n // 2]

print(f'Mediana campionaria: {x_mediana}')
```

```
Mediana campionaria: 117
```

Calcolo della moda campionaria

In [132…
```
max_f_x = 0
index = 0
count = 0

for i, f_i in enumerate(f_x):
    if f_i > max_f_x:
        max_f_x = f_i
        count = 1
        index = i
    elif f_i == max_f_x:
        count += 1

x_moda = v_x[index]
if count == 1:
    print(f'Moda campionaria unimodale: v_{index + 1} = {x_moda}, f_{index + 1} = {max_f_x}')
elif count == 2:
    print(f'Moda campionaria bimodale: v_{index + 1} = {x_moda}, f_{index + 1} = {max_f_x}')
else:
    print(f'Moda campionaria multimodale: v_{index + 1} = {x_moda}, f_{index + 1} = {max_f_x}')
```

```
Moda campionaria unimodale: v_4 = 105, f_4 = 10
```

Indici di variabilità

Calcolo della varianza campionaria

La varianza dei dati rispetto alla media campionaria è un po' alta.

In [133…
```
s2_x = 0
for x_i in x:
    s2_x += (x_i - x_mean) ** 2

s2_x = s2_x / (n - 1)
print(f'Varianza campionaria: {s2_x}')
```

```
Varianza campionaria: 132.59450800915334
```

Calcolo della deviazione standard campionaria

In [134…
```
s_x = sqrt(s2_x)
print(f'Deviazione standard campionaria: {s_x}')
```

```
Deviazione standard campionaria: 11.51496886705098
```

Calcolo dello scarto medio assoluto

In [135…
```
sa_x = 0
for x_i in x:
    sa_x += abs(x_i - x_mean)

sa_x = sa_x / n
print(f'Scarto medio assoluto: {sa_x}')
```

```
Scarto medio assoluto: 9.907296786389397
```

Calcolo dell'ampiezza del campo di variazione

In [136…
```
w_x = x[n - 1] - x[0]
print(f'Ampiezza del campo di variazione: {w_x}')
```

```
Ampiezza del campo di variazione: 38
```

Calcolo del coefficiente di variazione

In [137…
```
cv_x = s_x / x_mean
print(f'Coefficiente di variazione: {cv_x}')
```

```
Coefficiente di variazione: 0.09813409068555379
```

Indici di forma

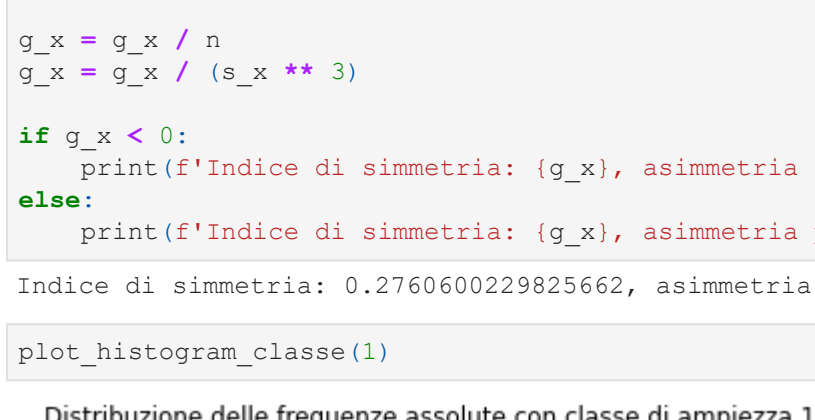
Calcolo dell'indice di simmetria

Dall'istogramma si nota una coda a destra, infatti l'indice è positivo.

In [138…
```
g_x = 0
for x_i in x:
    g_x += (x_i - x_mean) ** 3

g_x = g_x / n
g_x = g_x / (s_x ** 3)

if g_x < 0:
    print(f'Indice di simmetria: {g_x}, asimmetria negativa')
else:
    print(f'Indice di simmetria: {g_x}, asimmetria positiva')
```

```
Indice di simmetria: 0.2760600229825662, asimmetria positiva
```

In [139…
```
plot_histogram_classe(1)
```



Calcolo dell'indice di curtosi

In [140…
```
curtosi_x = 0
for x_i in x:
    curtosi_x += (x_i - x_mean) ** 4

curtosi_x = curtosi_x / n
curtosi_x = curtosi_x / (s_x ** 4)
curtosi_x = curtosi_x - 3

if curtosi_x > 0:
    print(f'Indice di curtosi: {curtosi_x}, è presente un eccesso di dati nelle classi centrali')
elif curtosi_x < 0:
    print(f'Indice di curtosi: {curtosi_x}, è presente una carenza di dati nelle classi centrali')
else:
    print(f'Indice di curtosi: {curtosi_x}, è presente una distribuzione di dati come quella di una distribuzio
```

```
Indice di curtosi: -1.196898991757091, è presente una carenza di dati nelle classi centrali
```

Quartili

Calcolo dei quartili

In [141…
```
def get_quartile(k):
    np = n * (k / 100)
    if np % 1 == 0:
        return (x[int(np)] + x[int(np) - 1]) / 2
    else:
        return x[int(np)]
```

In [142…
```
Q1_x = get_quartile(25)
Q2_x = get_quartile(50)
Q3_x = get_quartile(75)

print(f'Principali quartili: Q1 = {Q1_x}, Q2 = {Q2_x}, Q3 = {Q3_x}')
```

```
Principali quartili: Q1 = 105, Q2 = 117, Q3 = 126
```

Calcolo dello scarto interquartile

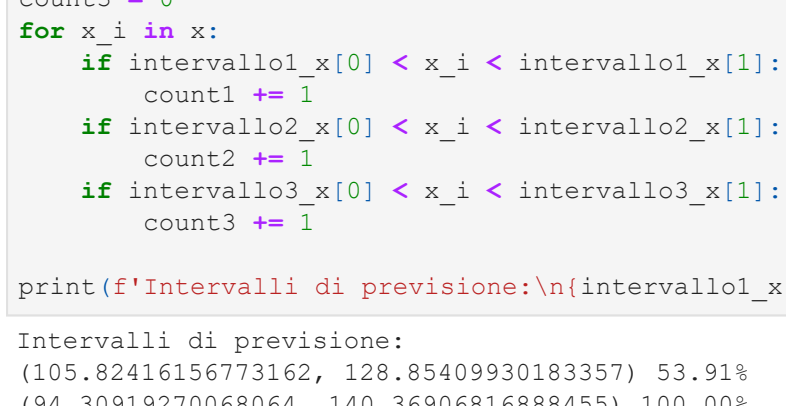
In [143…
```
si_x = Q3_x - Q1_x
print(f'Scarto interquartile: {si_x}')
```

```
Scarto interquartile: 21
```

Box plot

Come si evince dal grafico, non vi sono outliers. Se essi fosse presenti, sarebbero indicati nel grafico come dei punti rossi.

In [144…
```
fig, ax = plt.subplots()
ax.boxplot(x, flierprops=dict(marker='s', markerfacecolor='red'))
ax.set_title('Box plot')
ax.yaxis.set_ticks_position('none')
ax.grid(axis='y', linestyle='-', linewidth=0.5, color='lightgrey')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.get_xaxis().set_visible(False)
plt.show()
```



Intervalli di previsione

In [145…
```
intervallo1_x = (x_mean - s_x, x_mean + s_x)
intervallo2_x = (x_mean - 2 * s_x, x_mean + 2 * s_x)
intervallo3_x = (x_mean - 3 * s_x, x_mean + 3 * s_x)

count1 = 0
count2 = 0
count3 = 0
for x_i in x:
    if intervallo1_x[0] < x_i < intervallo1_x[1]:
        count1 += 1
    if intervallo2_x[0] < x_i < intervallo2_x[1]:
        count2 += 1
    if intervallo3_x[0] < x_i < intervallo3_x[1]:
        count3 += 1

print(f'Intervalli di previsione:\n{intervallo1_x} {(count1 / n) * 100:.2f}%\n{intervallo2_x} {(count2 / n) * 1
```

```
Intervalli di previsione:
(105.82416156773162, 128.85409930183357) 53.91%
(94.30919270068064, 140.36906816888455) 100.00%
(82.79422383362967, 151.88403703593553) 100.00%
```

Parsing dei dati delle calorie assunte

In [146…
```
with open('calorie.csv', 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    calorie = []
    reader.__next__()
    reader.__next__()
    for row in reader:
        calorie.append([row[3], float(row[8])])

df_calorie = pd.DataFrame(calorie, columns=['data', 'calorie'])
df_calorie['data'] = pd.to_datetime(df_calorie['data'])
df_calorie = df_calorie.groupby(pd.Grouper(key='data', freq='D')).sum().round(0)
df_calorie = df_calorie.reset_index()
df_calorie = df_calorie.groupby(pd.Grouper(key='data', freq='W')).mean().round(0)
df_calorie = df_calorie.reset_index()
print(f'Numero di ampiezza dei dati prima della eliminazione delle settimane in cui non vi sono state pesate co
df_calorie
```

```
Numero di ampiezza dei dati prima della eliminazione delle settimane in cui non vi sono state pesate corporee:
138
```

Out[146]:

	data	calorie
0	2019-10-20	2374.0
1	2019-10-27	1768.0
2	2019-11-03	1291.0
3	2019-11-10	1875.0
4	2019-11-17	1702.0
...	...	...
133	2022-05-08	2008.0
134	2022-05-15	1894.0
135	2022-05-22	1727.0
136	2022-05-29	1780.0
137	2022-06-05	1394.0

138 rows × 2 columns

In [147…
```
df_merge = pd.merge(df_peso, df_calorie, on='data', how='outer')
df_merge = df_merge.dropna()
df_merge = df_merge.reset_index()

print(f'Numero di ampiezza dei dati dopo l\'eliminazione delle settimane in cui non vi sono state pesate corpor
df_merge
```

```
Numero di ampiezza dei dati dopo l'eliminazione delle settimane in cui non vi sono state pesate corporee: 115
```

Out[147]:

	index	data	peso	calorie
0	0	2019-12-29	140.0	1798.0
1	2	2020-01-12	140.0	1816.0
2	4	2020-01-26	139.0	1813.0
3	5	2020-02-02	139.0	1922.0
4	8	2020-02-23	138.0	1538.0
...	...	...	...	...
110	123	2022-05-08	105.0	2008.0
111	124	2022-05-15	104.0	1894.0
112	125	2022-05-22	104.0	1727.0
113	126	2022-05-29	103.0	1780.0
114	127	2022-06-05	102.0	1394.0

115 rows × 4 columns

Dati bidimensionali

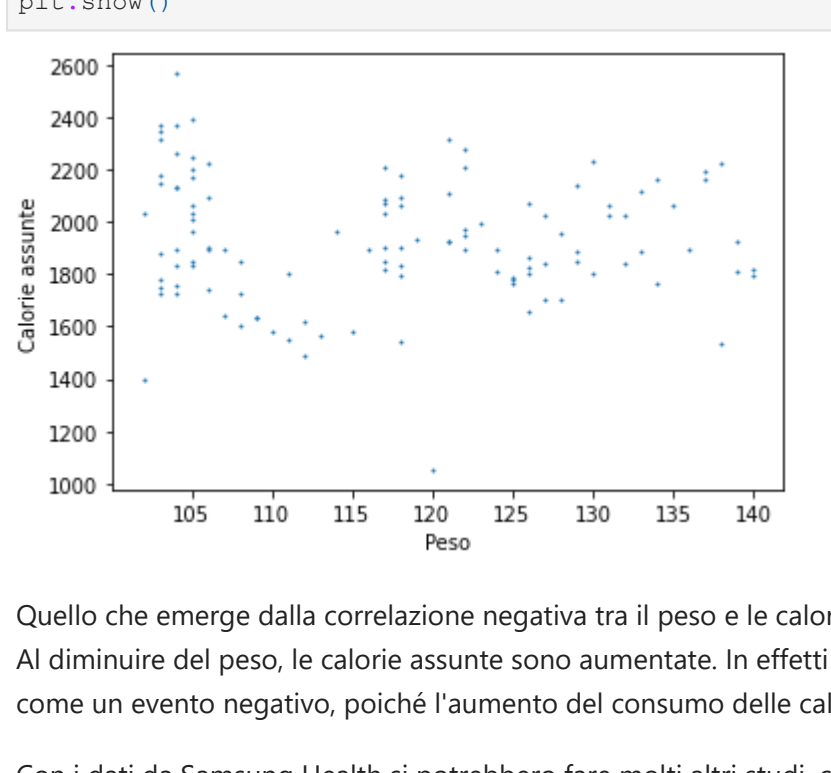
Calcolo dei coefficienti di correlazione campionario

In [148…
```
n = len(df_merge)
peso_mean = df_merge['peso'].mean()
calorie_mean = df_merge['calorie'].mean()
s_peso = df_merge['peso'].std()
s_calorie = df_merge['calorie'].std()

r = 0
for i in range(n):
    r += (df_merge['peso'][i] - peso_mean) * (df_merge['calorie'][i] - calorie_mean)
r = r / ((n - 1) * s_peso * s_calorie)

if r > 0:
    print(f'Coefficiente di correlazione campionario: {r}, è presente una correlazione positiva')
elif r < 0:
    print(f'Coefficiente di correlazione campionario: {r}, è presente una correlazione negativa')
else:
    print(f'Coefficiente di correlazione campionario: {r}, è presente una correlazione nulla')
```

```
Coefficiente di correlazione campionario: -0.06346445052282339, è presente una correlazione negativa
```

Diagramma a dispersione (scatter plot)

In [149…
```
fig, ax = plt.subplots()
ax.scatter(df_merge['peso'], df_merge['calorie'], s=1)
ax.set_xlabel('Peso')
ax.set_ylabel('Calorie assunte')

plt.show()
```



Quello che emerge dalla correlazione negativa tra il peso e le calorie è che quando il peso era alto, vi era un minore consumo di calorie. Al diminuire del peso, le calorie assunte sono aumentate. In effetti nella realtà questo accadde. Tuttavia, questo non è da interpretare come un evento negativo, poiché l'aumento del consumo delle calorie fu dato dalla maggiore attività fisica.

Con i dati da Samsung Health si potrebbero fare molti altri studi, come ad esempio quanto ha influito l'attività fisica sul peso, come è cambiato il battito cardiaco e la pressione sanguigna (la pressione scese di molto), ecc. Con l'incrocio dei dati si potrebbero fare tante cose interessanti.