Splitting Dataset - Outliers

Componenti del gruppo:

Justin Cadena
Francesco Miraglia
Zhou Zencheng

1 Splitting Dataset

[38]:
```
import pandas as pd

# Legge il file CSV e lo salva in un dataframe
df1 = pd.read_csv('Pokemon.csv')

# Mostra le prime righe del dataframe per verificare l'importazione
df1.head()
```

[38]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

1.1 Visione generale del dataframe

[39]:
```
print(df1.info())
print(df1.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
```

```
 0   #           800 non-null    int64
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      414 non-null    object
 4   Total       800 non-null    int64
 5   HP          800 non-null    int64
 6   Attack      800 non-null    int64
 7   Defense     800 non-null    int64
 8   Sp. Atk     800 non-null    int64
 9   Sp. Def     800 non-null    int64
 10  Speed       800 non-null    int64
 11  Generation  800 non-null    int64
 12  Legendary   800 non-null    bool
dtypes: bool(1), int64(9), object(3)
memory usage: 75.9+ KB
None
                #      Total          HP      Attack     Defense     Sp. Atk  \
count  800.000000  800.00000  800.000000  800.000000  800.000000  800.000000
mean   362.813750  435.10250   69.258750   79.001250   73.842500   72.820000
std    208.343798  119.96304   25.534669   32.457366   31.183501   32.722294
min      1.000000  180.00000    1.000000    5.000000    5.000000   10.000000
25%    184.750000  330.00000   50.000000   55.000000   50.000000   49.750000
50%    364.500000  450.00000   65.000000   75.000000   70.000000   65.000000
75%    539.250000  515.00000   80.000000  100.000000   90.000000   95.000000
max    721.000000  780.00000  255.000000  190.000000  230.000000  194.000000

          Sp. Def       Speed  Generation
count  800.000000  800.000000   800.00000
mean    71.902500   68.277500     3.32375
std     27.828916   29.060474     1.66129
min     20.000000    5.000000     1.00000
25%     50.000000   45.000000     2.00000
50%     70.000000   65.000000     3.00000
75%     90.000000   90.000000     5.00000
max    230.000000  180.000000     6.00000
```

1.2 Filtraggio dei dati

[40]:
```
# Seleziona una singola colonna
print(df1['Name'])

# Seleziona righe basate su condizioni
print(df1[df1['HP'] > 150])
```

```
0                  Bulbasaur
1                    Ivysaur
2                   Venusaur
3      VenusaurMega Venusaur
```

```
4                 Charmander
               ...
795                  Diancie
796       DiancieMega Diancie
797       HoopaHoopa Confined
798        HoopaHoopa Unbound
799                 Volcanion
Name: Name, Length: 800, dtype: object
       #       Name   Type 1 Type 2  Total   HP  Attack  Defense  Sp. Atk  \
121  113    Chansey   Normal    NaN    450  250       5        5       35
155  143    Snorlax   Normal    NaN    540  160     110       65       65
217  202  Wobbuffet  Psychic    NaN    405  190      33       58       33
261  242    Blissey   Normal    NaN    540  255      10       10       75
351  321    Wailord    Water    NaN    500  170      90       45       90
655  594  Alomomola    Water    NaN    470  165      75       80       40

     Sp. Def  Speed  Generation  Legendary
121      105     50           1      False
155      110     30           1      False
217       58     33           2      False
261      135     55           2      False
351       45     60           3      False
655       45     65           5      False
```

[41]:
```
import pandas as pd

df = pd.read_csv('Serie A.csv')

df.head()
```

[41]:

	Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	...	\
0	I1	18/08/2018	Chievo	Juventus	2	3	A	1	1	D	...	
1	I1	18/08/2018	Lazio	Napoli	1	2	A	1	1	D	...	
2	I1	19/08/2018	Bologna	Spal	0	1	A	0	0	D	...	
3	I1	19/08/2018	Empoli	Cagliari	2	0	H	1	0	H	...	
4	I1	19/08/2018	Parma	Udinese	2	2	D	1	0	H	...	

	BbAv<2.5	BbAH	BbAHh	BbMxAHH	BbAvAHH	BbMxAHA	BbAvAHA	PSCH	PSCD	\
0	2.13	19	2.00	1.68	1.64	2.38	2.29	18.84	6.42	
1	2.17	20	0.00	2.12	2.07	1.83	1.79	2.78	3.57	
2	1.58	19	-0.25	1.97	1.92	1.99	1.94	2.31	3.18	
3	1.71	19	-0.25	1.98	1.91	1.98	1.94	2.54	3.42	
4	1.65	20	0.00	1.81	1.77	2.18	2.10	2.80	3.24	

	PSCA
0	1.22
1	2.59

```
2  3.59
3  2.95
4  2.78

[5 rows x 61 columns]
```

[42]:
```
import csv

# Apre il file CSV e lo legge
with open('Serie A.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

```
  Div        Date HomeTeam  AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  ...  \
0  I1  18/08/2018   Chievo  Juventus     2     3   A     1     1   D  ...
1  I1  18/08/2018    Lazio    Napoli     1     2   A     1     1   D  ...
2  I1  19/08/2018  Bologna      Spal     0     1   A     0     0   D  ...
3  I1  19/08/2018   Empoli  Cagliari     2     0   H     1     0   H  ...
4  I1  19/08/2018    Parma   Udinese     2     2   D     1     0   H  ...

 BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA   PSCH  PSCD  \
0      2.13    19   2.00     1.68     1.64     2.38     2.29  18.84  6.42
1      2.17    20   0.00     2.12     2.07     1.83     1.79   2.78  3.57
2      1.58    19  -0.25     1.97     1.92     1.99     1.94   2.31  3.18
3      1.71    19  -0.25     1.98     1.91     1.98     1.94   2.54  3.42
4      1.65    20   0.00     1.81     1.77     2.18     2.10   2.80  3.24

   PSCA
0  1.22
1  2.59
2  3.59
3  2.95
4  2.78
```

[43]:
```
import numpy as np

# Carica il file CSV in un array con numpy
data = np.genfromtxt('Pokemon.csv', delimiter=',')

# Stampiamo i primi 15 elementi per verificare l'importazione
print(data[:20])
```

```
[[  1.  nan  nan  nan 318.  45.  49.  49.  65.  65.  45.   1.  nan]
 [  2.  nan  nan  nan 405.  60.  62.  63.  80.  80.  60.   1.  nan]
 [  3.  nan  nan  nan 525.  80.  82.  83. 100. 100.  80.   1.  nan]
 [  3.  nan  nan  nan 625.  80. 100. 123. 122. 120.  80.   1.  nan]
```

```
[  4.  nan  nan  nan 309.  39.  52.  43.  60.  50.  65.   1.  nan]
[  5.  nan  nan  nan 405.  58.  64.  58.  80.  65.  80.   1.  nan]
[  6.  nan  nan  nan 534.  78.  84.  78. 109.  85. 100.   1.  nan]
[  6.  nan  nan  nan 634.  78. 130. 111. 130.  85. 100.   1.  nan]
[  6.  nan  nan  nan 634.  78. 104.  78. 159. 115. 100.   1.  nan]
[  7.  nan  nan  nan 314.  44.  48.  65.  50.  64.  43.   1.  nan]
[  8.  nan  nan  nan 405.  59.  63.  80.  65.  80.  58.   1.  nan]
[  9.  nan  nan  nan 530.  79.  83. 100.  85. 105.  78.   1.  nan]
[  9.  nan  nan  nan 630.  79. 103. 120. 135. 115.  78.   1.  nan]
[ 10.  nan  nan  nan 195.  45.  30.  35.  20.  20.  45.   1.  nan]
[ 11.  nan  nan  nan 205.  50.  20.  55.  25.  25.  30.   1.  nan]
[ 12.  nan  nan  nan 395.  60.  45.  50.  90.  80.  70.   1.  nan]
[ 13.  nan  nan  nan 195.  40.  35.  30.  20.  20.  50.   1.  nan]
[ 14.  nan  nan  nan 205.  45.  25.  50.  25.  25.  35.   1.  nan]
[ 15.  nan  nan  nan 395.  65.  90.  40.  45.  80.  75.   1.  nan]
```

---

2 Outliers

[66]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Crea un DataFrame di esempio
data = {'Valori': [1, 2, 3, 4, 5, 10, 15, 20, 25, 300, 1000, 100000000,␣
 ↪-50000000, -50]}
df = pd.DataFrame(data)
# Lista con outliers da entrambi i lati

# Calcola la media e la deviazione standard
mean_value = df['Valori'].mean()
std_dev = df['Valori'].std()

# Identifica gli outliers considerando ±3 sigma dalla media
outliers = df[(df['Valori'] > mean_value + 3 * std_dev) | (df['Valori'] <␣
 ↪mean_value - 3 * std_dev)]
outliers
```

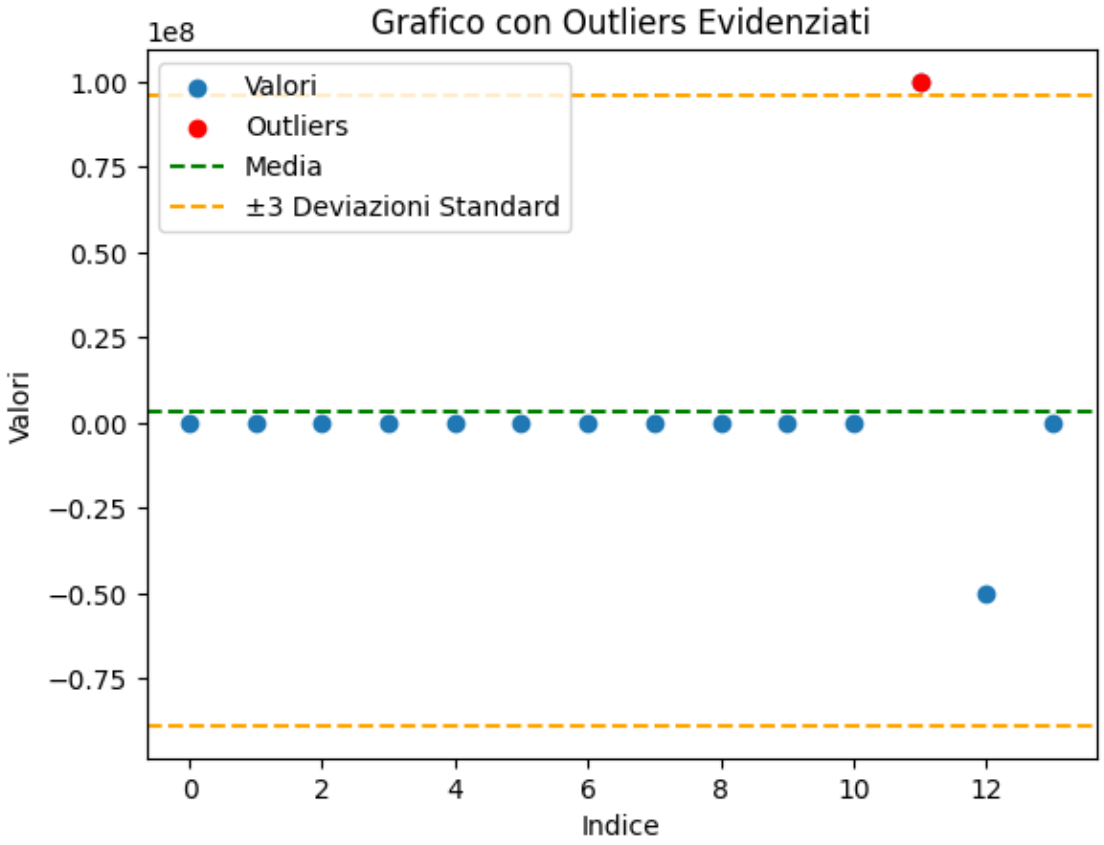
[66]:

	Valori
11	100000000

3 Grafico a dispersione

[67]:
```
# Crea un grafico a dispersione
plt.scatter(df.index, df['Valori'], label='Valori')

# Evidenzia gli outliers nel grafico con un colore diverso
```

```
plt.scatter(outliers.index, outliers['Valori'], color='red', label='Outliers')

# Aggiungi la media e la deviazione standard al grafico
plt.axhline(y=mean_value, color='green', linestyle='--', label='Media')
plt.axhline(y=mean_value + 3 * std_dev, color='orange', linestyle='--',␣
 ↪label='±3 Deviazioni Standard')
plt.axhline(y=mean_value - 3 * std_dev, color='orange', linestyle='--')

# Aggiungi etichette e legenda al grafico
plt.xlabel('Indice')
plt.ylabel('Valori')
plt.title('Grafico con Outliers Evidenziati')
plt.legend()

# Mostra il grafico
plt.show()
```



4 Metodo Z-score

[68]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Crea un DataFrame di esempio con 4 features
data = {'Feature1': [1, 2000, 3, 4, 50000, 10, 15, 20, 2500000, 300000000,␣
 ↪100000000],
        'Feature2': [2, 4, 6, 8, 10, 20, 30, 40, 50000, 60, 200],
        'Feature3': [5, 10, 15, 20000, 25, 50, 75, 100, 125, 150, 500000],
        'Feature4': [1, -20000000, 3, 4000000000, 5, 10, 15, 20, 20005, 30,␣
 ↪10000]}

df = pd.DataFrame(data)

# Definisci il numero minimo di features che devono superare la soglia per␣
 ↪considerare un dato un outlier
min_features_threshold = 1
k=2 #intervallo di confidenza

# Lista per salvare gli indici degli outliers
outlier_indices = []

# Itera su ogni feature
for feature in df.columns:
    mean_value = df[feature].mean()
    std_dev = df[feature].std()

    # Identifica gli outliers per ciascuna feature
    df['Outlier_' + feature] = (df[feature] > mean_value + k * std_dev) |␣
 ↪(df[feature] < mean_value - k * std_dev)
df
```

[68]:

	Feature1	Feature2	Feature3	Feature4	Outlier_Feature1	\
0	1	2	5	1	False	
1	2000	4	10	-20000000	False	
2	3	6	15	3	False	
3	4	8	20000	4000000000	False	
4	50000	10	25	5	False	
5	10	20	50	10	False	
6	15	30	75	15	False	
7	20	40	100	20	False	
8	2500000	50000	125	20005	False	
9	300000000	60	150	30	True	
10	100000000	200	500000	10000	False	

	Outlier_Feature2	Outlier_Feature3	Outlier_Feature4

0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	True
4	False	False	False
5	False	False	False
6	False	False	False
7	False	False	False
8	True	False	False
9	False	False	False
10	False	True	False

5 Identificazione e marcatura dei record outliers nel DataFrame

[69]:
```
# Calcola il numero di features che superano la soglia per ogni riga
df['Num_Outliers'] = df.filter(like='Outlier_').sum(axis=1)

# Filtra i dati per mantenere solo le righe con almeno il numero minimo di␣
 ↪features superanti la soglia
outliers = df[df['Num_Outliers'] >= min_features_threshold]

# Aggiungi una colonna che indica se il record è un outlier o meno
df['Is_Outlier'] = df.index.isin(outliers.index)

# Rimuovi colonne ausiliarie
df.drop(df.filter(like='Outlier_').columns, axis=1, inplace=True)
df.drop('Num_Outliers', axis=1, inplace=True)
df
```

[69]:

	Feature1	Feature2	Feature3	Feature4	Is_Outlier
0	1	2	5	1	False
1	2000	4	10	-20000000	False
2	3	6	15	3	False
3	4	8	20000	4000000000	True
4	50000	10	25	5	False
5	10	20	50	10	False
6	15	30	75	15	False
7	20	40	100	20	False
8	2500000	50000	125	20005	True
9	300000000	60	150	30	True
10	100000000	200	500000	10000	True

6 Calcolo delle features che superano la soglia

[70]:
```
#Calcola il numero di features che superano la soglia per ogni riga
df['Num_Outliers'] = df.filter(like='Outlier_.').sum(axis=1)
df
```

[70]:

	Feature1	Feature2	Feature3	Feature4	Is_Outlier	Num_Outliers
0	1	2	5	1	False	0.0
1	2000	4	10	-20000000	False	0.0
2	3	6	15	3	False	0.0
3	4	8	20000	4000000000	True	0.0
4	50000	10	25	5	False	0.0
5	10	20	50	10	False	0.0
6	15	30	75	15	False	0.0
7	20	40	100	20	False	0.0
8	2500000	50000	125	20005	True	0.0
9	300000000	60	150	30	True	0.0
10	100000000	200	500000	10000	True	0.0

7 Filtraggio dei dati che mantengono il numero di features che superano la soglia

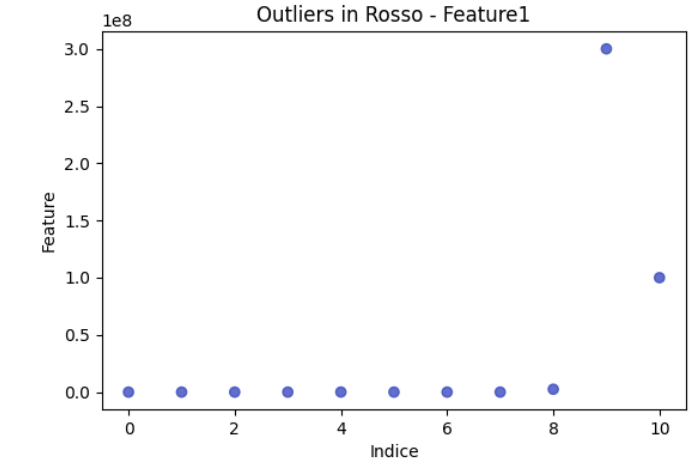
[71]:
```
# Filtra i dati per mantenere solo le righe con almeno il numero minimo di␣
 ↪features superanti la soglia
outliers = df[df['Num_Outliers'] >= min_features_threshold]

# Aggiungi una colonna che indica se il record è un outlier o meno
df['Is_Outlier'] = df.index.isin(outliers.index)
# Rimuovi colonne ausiliarie
df.drop(df.filter(like='Outlier_').columns, axis=1, inplace=True)
df.drop('Num_Outliers', axis=1, inplace=True)
df
```

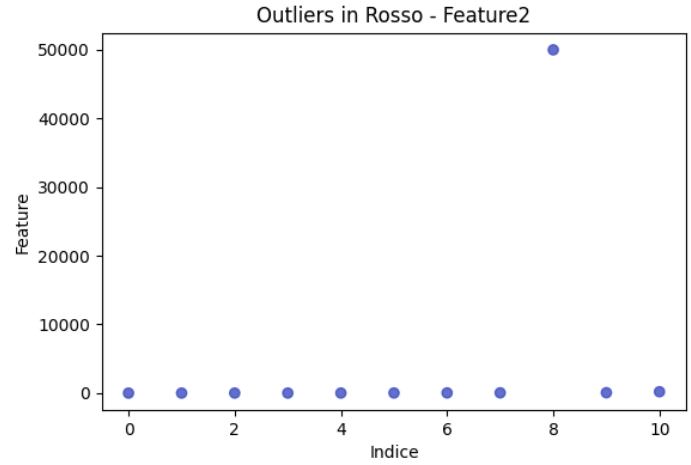
[71]:

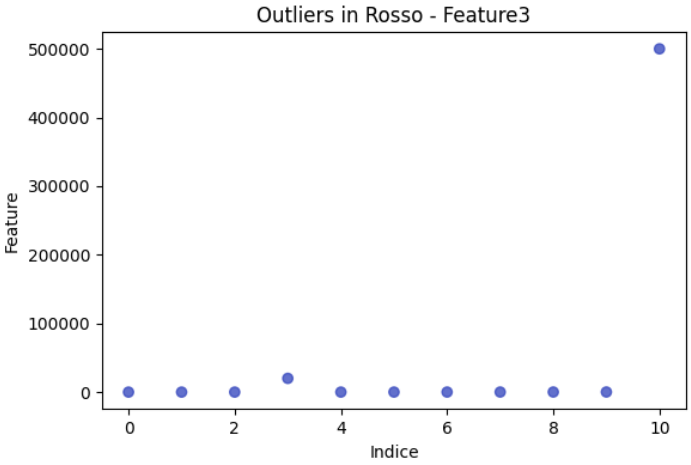
	Feature1	Feature2	Feature3	Feature4	Is_Outlier
0	1	2	5	1	False
1	2000	4	10	-20000000	False
2	3	6	15	3	False
3	4	8	20000	4000000000	False
4	50000	10	25	5	False
5	10	20	50	10	False
6	15	30	75	15	False
7	20	40	100	20	False
8	2500000	50000	125	20005	False
9	300000000	60	150	30	False
10	100000000	200	500000	10000	False

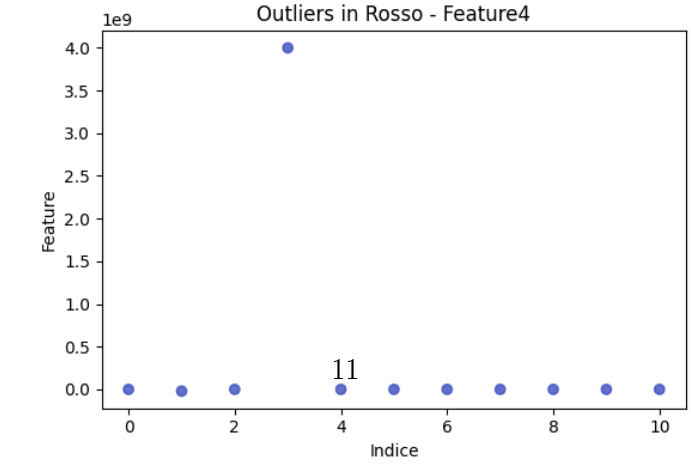
8 Grafico a matrice

[72]:
```
# Organizza i grafici in una matrice, con una colonna e 4 righe
num_features = len(df.columns) - 1  # Escludi la colonna 'Is_Outlier'
num_rows = num_features
num_cols = 1  # Una colonna

plt.figure(figsize=(6, 4 * num_rows))
for i, feature in enumerate(df.columns[:-1]):  # Escludi la colonna 'Is_Outlier'
    plt.subplot(num_rows, num_cols, i + 1)
    plt.scatter(df.index, df[feature], c=df['Is_Outlier'], cmap='coolwarm',␣
 ↪alpha=0.8)
    plt.title(f'Outliers in Rosso - {feature}')
    plt.xlabel('Indice')
    plt.ylabel('Feature')

plt.tight_layout()
plt.show()
```


Outliers in Rosso - Feature1
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Feature
0
2
4
6
8
10
Indice


Outliers in Rosso - Feature2
50000
40000
30000
20000
10000
0
Feature
0
2
4
6
8
10
Indice


Outliers in Rosso - Feature3
500000
400000
300000
200000
100000
0
Feature
0
2
4
6
8
10
Indice


Outliers in Rosso - Feature4
1e9
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Feature
0
2
4
6
8
10
Indice

[73]:
```
#Elimina le righe corrispondenti agli outliers quelli che hanno almeno una␣
↪feature fuori scala
df_filtered = df[df['Is_Outlier'] == False]
df_filtered
```

[73]:

	Feature1	Feature2	Feature3	Feature4	Is_Outlier
0	1	2	5	1	False
1	2000	4	10	-20000000	False
2	3	6	15	3	False
3	4	8	20000	4000000000	False
4	50000	10	25	5	False
5	10	20	50	10	False
6	15	30	75	15	False
7	20	40	100	20	False
8	2500000	50000	125	20005	False
9	300000000	60	150	30	False
10	100000000	200	500000	10000	False

9 Deviazione standard

[74]:
```
def calcola_deviazione_standard(lista):
    n = len(lista)

    # Calcola la media
    media = sum(lista) / n

    # Calcola la somma dei quadrati delle differenze dalla media
    somma_quadrati_diff = sum((x - media) ** 2 for x in lista)

    # Calcola la deviazione standard
    deviazione_standard = (somma_quadrati_diff / n) ** 0.5

    return deviazione_standard

# Esempio di utilizzo
numero_lista = [1, 2, 3, 4, 5]
deviazione_standard = calcola_deviazione_standard(numero_lista)

# Stampa il risultato
print(f"La deviazione standard della lista è: {deviazione_standard}")
```

```
La deviazione standard della lista è: 1.4142135623730951
```