E adesso IA

I dataset

Un dataset è un insieme strutturato di dati che è organizzato in righe e colonne, simile a una tabella.

perchè è importante fare i grafici?

fare i grafici è importante per poter capire i dati a colpo d'occhio, dove altrimenti ci si dovrebbe fermare a leggere

Missing Values

I missing value sono valori che vengono persi nel trasferimento di dati, non registrati o che sono in generale mancanti.

possono essere gestiti in diversi modi:

1. eliminazione
2. riempimento

   eliminazione, semplicemente andiamo ad eliminare l'intera riga in cui è presente un valore mancante (non è largamente utilizzata per l'enorme spreco di dati che genera)

```
import pandas as pd

# Dataset con dati mancanti rappresentati da None o NaN
dataset = [
    {"età": 25, "punteggio": 90, "ammesso": 1},
    {"età": None, "punteggio": 85, "ammesso": 0},
    {"età": 28, "punteggio": None, "ammesso": 1},
    {"età": None, "punteggio": 75, "ammesso": 1},
    {"età": 23, "punteggio": None, "ammesso": None},
    {"età": 23, "punteggio": 77, "ammesso": None},
]
df = pd.DataFrame(dataset) #dataframe è una tabella
df
```

```
    età  punteggio  ammesso
0  25.0       90.0      1.0
1   NaN       85.0      0.0
2  28.0        NaN      1.0
3   NaN       75.0      1.0
4  23.0        NaN      NaN
5  23.0       77.0      NaN
```

1. Dataframe creato

   (normalmente si tende ad importarli dall'esterno)

```
df['punteggio']
```

```
0    90.0
1    85.0
2     NaN
3    75.0
4     NaN
5    77.0
Name: punteggio, dtype: float64
```

```
#identificazione delle righe con dati mancanti
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

```
    età  punteggio  ammesso
1   NaN       85.0      0.0
2  28.0        NaN      1.0
3   NaN       75.0      1.0
4  23.0        NaN      NaN
5  23.0       77.0      NaN
```

```
#Conta quante righe con dati mancanti ci sono in totale
totale_dati_mancanti = righe_con_dati_mancanti.shape[0] #0=righe,
1=colonne
totale_dati_mancanti
```

```
5
```

Mostriamo il totale dei dati mancanti

```
print('righe con dati mancanti:')
print(righe_con_dati_mancanti)
print('Totale dati mancanti: ',totale_dati_mancanti)
```

```
righe con dati mancanti:
    età  punteggio  ammesso
1   NaN       85.0      0.0
2  28.0        NaN      1.0
3   NaN       75.0      1.0
4  23.0        NaN      NaN
5  23.0       77.0      NaN
Totale dati mancanti:  5
```

```
import pandas as pd

# Dataset con dati mancanti rappresentati da None o NaN
dataset = [
```

```
    {"nome": "Alice", "età": 25, "punteggio": 90, "email":
"alice@email.com"},
    {"nome": "Bob", "età": 22, "punteggio": None, "email": None},
    {"nome": "Charlie", "età": 28, "punteggio": 75, "email":
"charlie@email.com"},
]

# Converti il dataset in un DataFrame
df = pd.DataFrame(dataset)
df
```

```
      nome  età  punteggio              email
0    Alice   25       90.0    alice@email.com
1      Bob   22        NaN               None
2  Charlie   28       75.0  charlie@email.com
```

1. Prendiamo il dataframe e rimpiazziamone le righe sbagliate
   - creiamo un secondo dataframe senza righe sbagliate

```
#Rimuovi le righe con dati mancanti
df1=df.dropna(inplace=False) #crea un nuovo dataframe in questo caso
senza riga
                             #se è true allora sostituisce il nuovo
dataframe in generale con quello nuovo
df1
```

```
      nome  età  punteggio              email
0    Alice   25       90.0    alice@email.com
2  Charlie   28       75.0  charlie@email.com
```

Il metodo per riempimento riempe con il valore medio i dati mancanti

1. andiamo a creare un dataframe e importiamo le librerie necessarie per l'analisi dei dati

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

# Genera dati di esempio
data = {
    'Variable1': [1, 2, 3, 4, 5],
    'Variable2': [1, 2, np.nan, 4, np.nan],
    'Missing_Column': ['A', 'B', 'A', 'C', np.nan]
}
# Crea un DataFrame
df = pd.DataFrame(data)
df1=pd.DataFrame()
df
```

```
   Variable1  Variable2 Missing_Column
0          1        1.0              A
1          2        2.0              B
2          3        NaN              A
3          4        4.0              C
4          5        NaN            NaN
```

1. Seleziona le colonne contenenti variabili numeriche

```
#trattamento dei missing values nelle variabili numeriche
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols
```

```
   Variable1  Variable2
0          1        1.0
1          2        2.0
2          3        NaN
3          4        4.0
4          5        NaN
```

- mostra le colonne con dati numerici

```
#trattamento dei missing values nelle variabili numeriche
numeric_cols = df.select_dtypes(include=['number']) #include colonne numeriche e fa vedere solo categoriche
numeric_cols.columns
```

```
Index(['Variable1', 'Variable2'], dtype='object')
```

- sostituisce i valori mancanti con la media e li assegna ad un nuovo dataframe

  assegnare ad un nuovo dataframe è utile per avere quello originale sempre a portata di mano qual'ora sostituissimo o danneggiassimo dati importanti nel processo

```
df1[numeric_cols.columns] =
df[numeric_cols.columns].fillna(df[numeric_cols.columns].mean())
df1

#crea colonne con stessi nomi (var1 e var2) = assegna valori delle colonne df originale(mean=media)
```

```
   Variable1  Variable2
0          1   1.000000
1          2   2.000000
2          3   2.333333
3          4   4.000000
4          5   2.333333
```

1. trattamento variabili numeriche e categoriche

- asseggniamo le variabili categoriche ed esclude i datatype "numbers" (ovvero numeri)

```
#trattamento dei missing values nelle variabili categoriche
categorical_cols = df.select_dtypes(exclude=['number']) #esclude colonne numeriche e fa vedere solo categoriche
categorical_cols.columns
```

```
Index(['Missing_Column'], dtype='object')
```

- sostituisce i valori mancanti con la media e li assegna ad un nuovo dataframe

  assegnare ad un nuovo dataframe è utile per avere quello originale sempre a portata di mano qual'ora sostituissimo o danneggiassimo dati importanti nel processo

```
df1[numeric_cols.columns] = df[numeric_cols.columns].fillna(df[numeric_cols.columns].mean().iloc[1])
df1
#iloc[] = gestisce indici
```

```
   Variable1  Variable2
0          1   1.000000
1          2   2.000000
2          3   2.333333
3          4   4.000000
4          5   2.333333
```

Stampa il dataframe

```
print(f"il primo coi valori mancati \n{df} \ne")
```

```
il primo coi valori mancati
   Variable1  Variable2 Missing_Column
0          1        1.0              A
1          2        2.0              B
2          3        NaN              A
3          4        4.0              C
4          5        NaN            NaN
e
```

Analisi dei dati

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

# Genera dati di esempio
data = {
    'Feature1': [1, 2, np.nan, 4, 5],
    'Feature2': [np.nan, 2, 3, 4, np.nan],
```

```
    'Feature3': [1, np.nan, 3, 4, 5]
}
# Crea un DataFrame
df = pd.DataFrame(data)
df
```

```
   Feature1  Feature2  Feature3
0       1.0       NaN       1.0
1       2.0       2.0       NaN
2       NaN       3.0       3.0
3       4.0       4.0       4.0
4       5.0       NaN       5.0
```

1. Creiamo un dataframe

```
df.isnull()
```

```
   Feature1  Feature2  Feature3
0     False      True     False
1     False     False      True
2      True     False     False
3     False     False     False
4     False      True     False
```

1. Vediamo quanti sono i valori mancanti per ogni colonna

```
df.isnull().sum()
```

```
Feature1    1
Feature2    2
Feature3    1
dtype: int64
```

1. calcoliamo la percentuale di essi

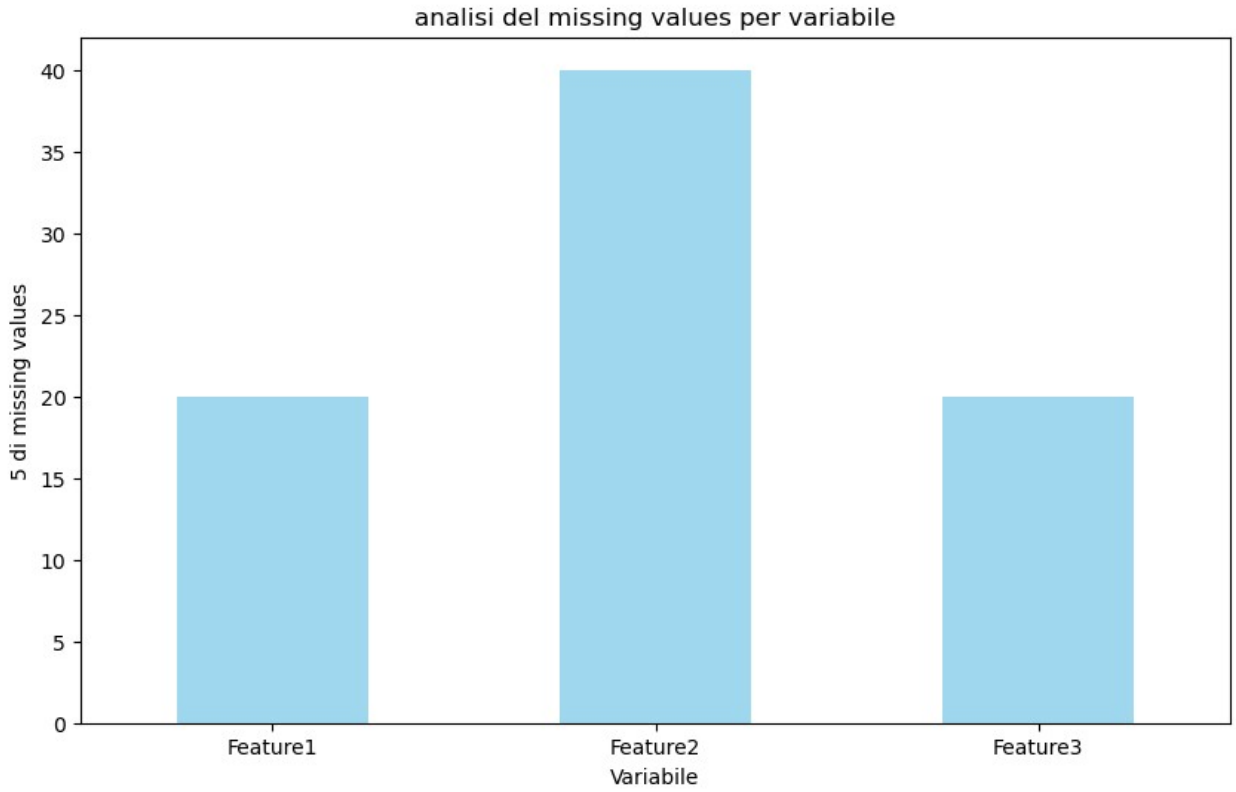
```
missing_percent = df.isnull().sum() / len(df) * 100
missing_percent
```

```
Feature1    20.0
Feature2    40.0
Feature3    20.0
dtype: float64
```

1. Mostriamo questa percentuale su un grafico a barre

```
#Calcola la percentuale di righe con missing values per ciascuna
variabile
missing_percent= (df.isnull().sum()) / len(df) * 100

#crea il grafico a barre
plt.figure(figsize=(10,6))
```

```
missing_percent.plot(kind='bar', color='skyblue', alpha=0.8)
plt.xlabel('Variabile')
plt.ylabel('5 di missing values')
plt.title('analisi del missing values per variabile')
plt.xticks(rotation=0)
plt.show()
```



le heatmap

le heatmap sono usate per avere una visione a matrice dei dati mancanti, anche molto utile per sapere a "colpo dd'occhio" quanto il dataframe sia completo e di qualità

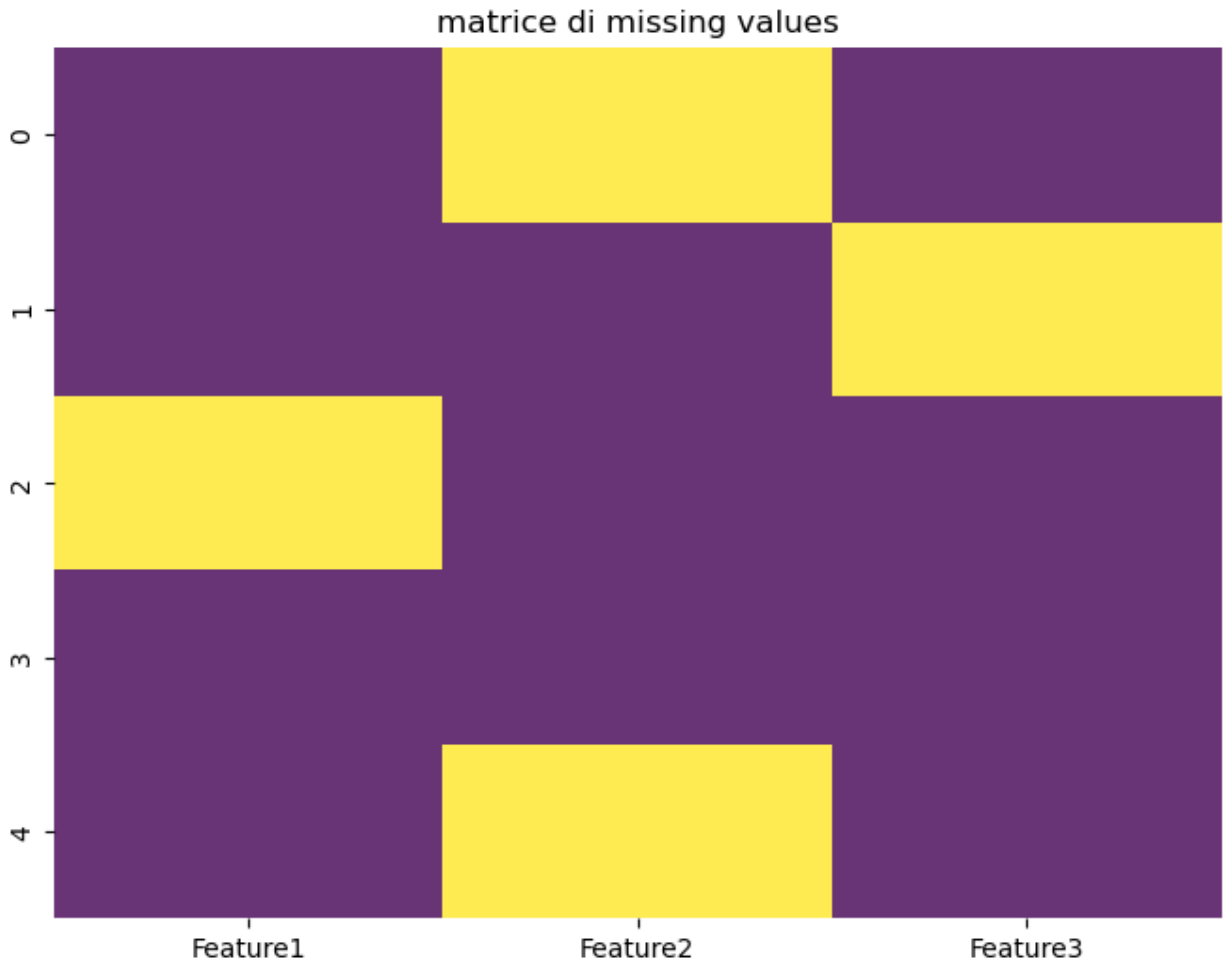
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Genera dati di esempio
data = {
    'Feature1': [1, 2, np.nan, 4, 5],
    'Feature2': [np.nan, 2, 3, 4, np.nan],
    'Feature3': [1, np.nan, 3, 4, 5]
}

# Crea un DataFrame
```

```
df = pd.DataFrame(data)

# Calcola la matrice di missing values
missing_matrix = df.isnull()
missing_matrix
```

	Feature1	Feature2	Feature3
0	False	True	False
1	False	False	True
2	True	False	False
3	False	False	False
4	False	True	False

1. Dopo la creazione o importazione di un dataframe andiamo a creare un'heatmap utilizzando seaborn

```
#crea heatmap colorata
plt.figure(figsize=(8,6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
plt.title('matrice di missing values')
plt.show()
```



1. asseggnamo dei valori e una palette a questa heatmap

```
plt.figure(figsize=(8,6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=True,alpha=0.8)
plt.title('matrice di missing values')
plt.show()
```


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Genera dati casuali per l'esplorazione
np.random.seed(42)
data = {
    'Età': np.random.randint(18, 70, size=1000),
    'Genere': np.random.choice(['Maschio', 'Femmina'], size=1000),
    'Punteggio': np.random.uniform(0, 100, size=1000),
    'Reddito': np.random.normal(50000, 15000, size=1000)
#distribuzione gaussiana
}

df = pd.DataFrame(data)
```

```python
# Visualizza le prime righe del dataset
print(df.head())
```

```
   Età   Genere  Punteggio       Reddito
0   56  Maschio  85.120691  52915.764524
1   69  Maschio  49.514653  44702.505608
2   46  Maschio  48.058658  55077.257652
3   32  Femmina  59.240778  45568.978848
4   60  Maschio  82.468097  52526.914644
```

```python
#informazioni sul dataset
print(df.info())
```

```python
#statistiche desrittive
print(df.describe())
```

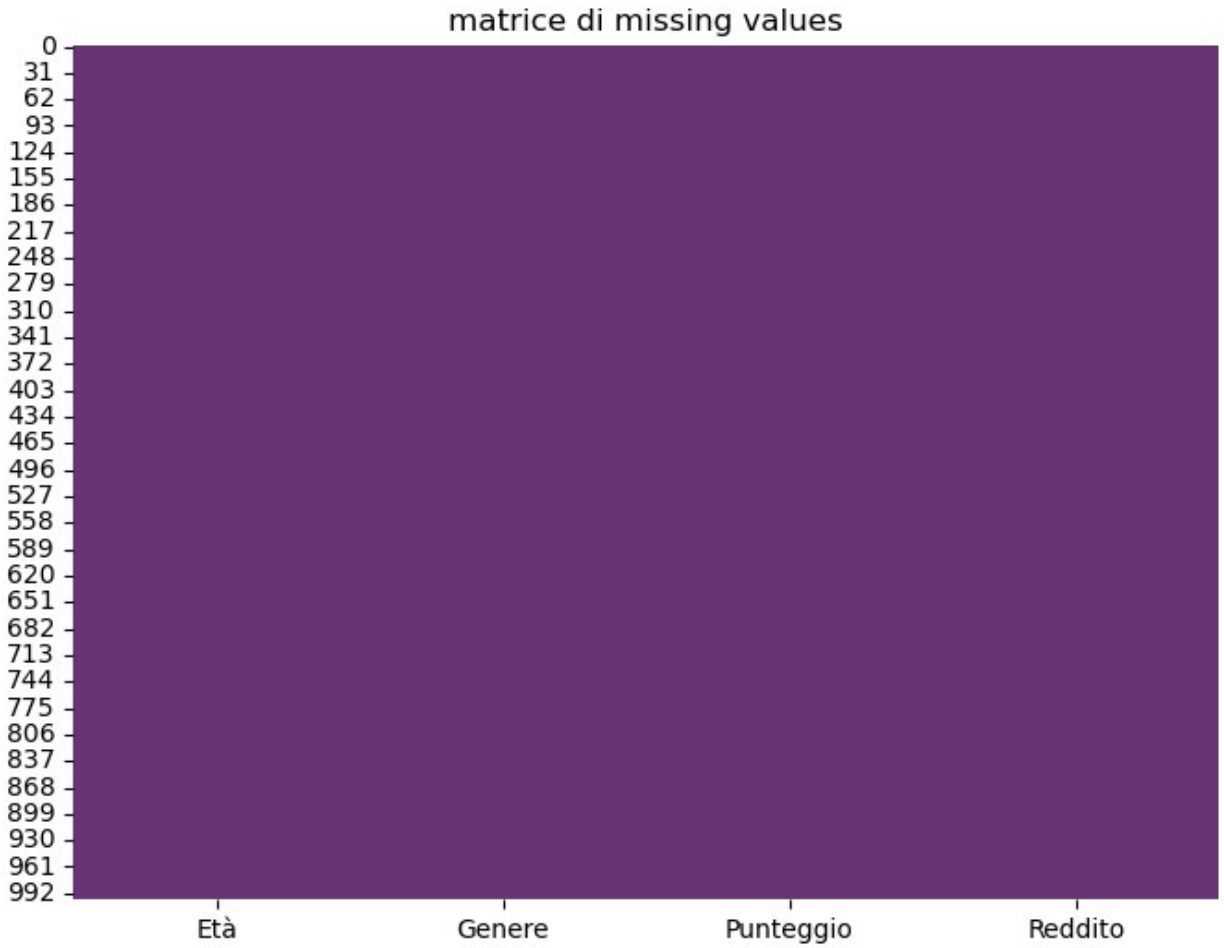
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Età        1000 non-null   int64
 1   Genere     1000 non-null   object
 2   Punteggio  1000 non-null   float64
 3   Reddito    1000 non-null   float64
dtypes: float64(2), int64(1), object(1)
memory usage: 31.4+ KB
None
               Età    Punteggio       Reddito
count  1000.00000  1000.000000   1000.000000
mean     43.81900    50.471078  50241.607607
std      14.99103    29.014970  14573.000585
min      18.00000     0.321826   4707.317663
25%      31.00000    24.690382  40538.177863
50%      44.00000    51.789520  50099.165858
75%      56.00000    75.549365  60089.683773
max      69.00000    99.941373  97066.228005
```

```python
#gesione valori mancanti
missing_data = df.isnull().sum()
print('valori mancanti per ciascuna colonna: ')
print(missing_data)
```

```
valori mancanti per ciascuna colonna:
Età          0
Genere       0
Punteggio    0
Reddito      0
dtype: int64
```

il procedimento è lo stesso del precedente, tranne che in questo caso non ci sono valori mancanti

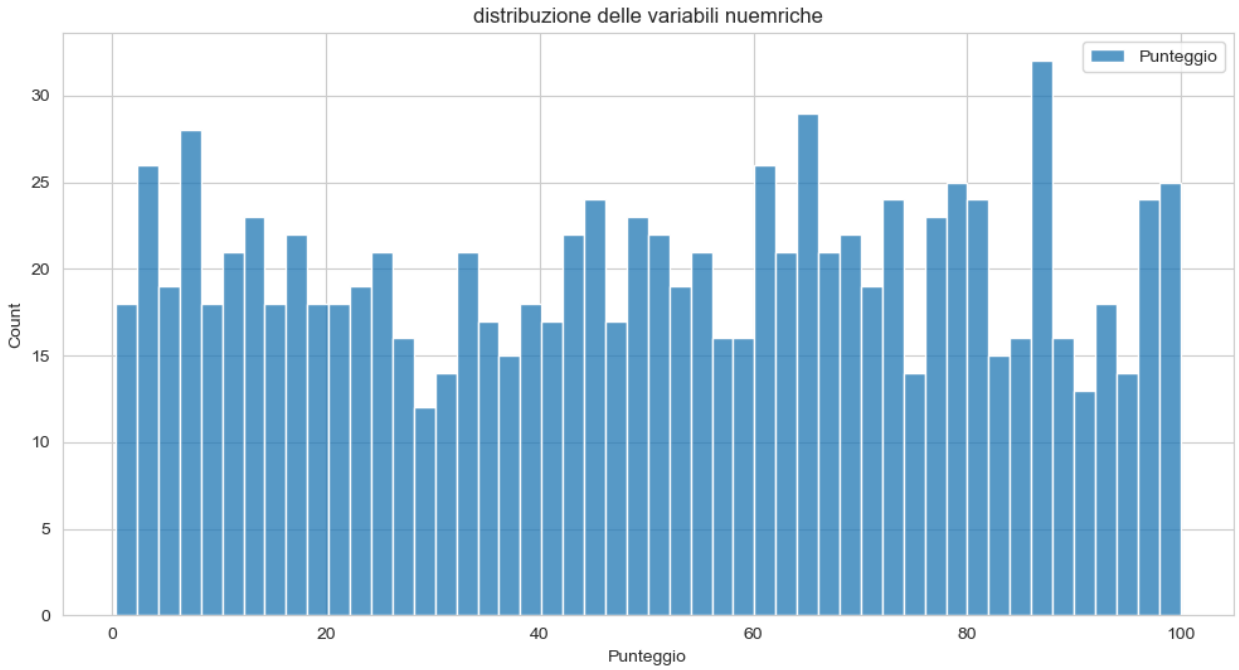
```
plt.figure(figsize=(8,6))
sns.heatmap(df.isnull(), cmap='viridis', cbar=False,alpha=0.8)
plt.title('matrice di missing values')
plt.show()
```



visualizzazione della distribuzione

tramite istogramma

```
#visualizza la distribuzione dele varibaili numeriche

plt.figure(figsize=(12,6))
sns.set_style('whitegrid')
sns.histplot(df['Punteggio'], kde=False, bins=50, label='Punteggio')
#istogramma
plt.legend()
plt.title('distribuzione delle variabili nuemriche')
plt.show()
```



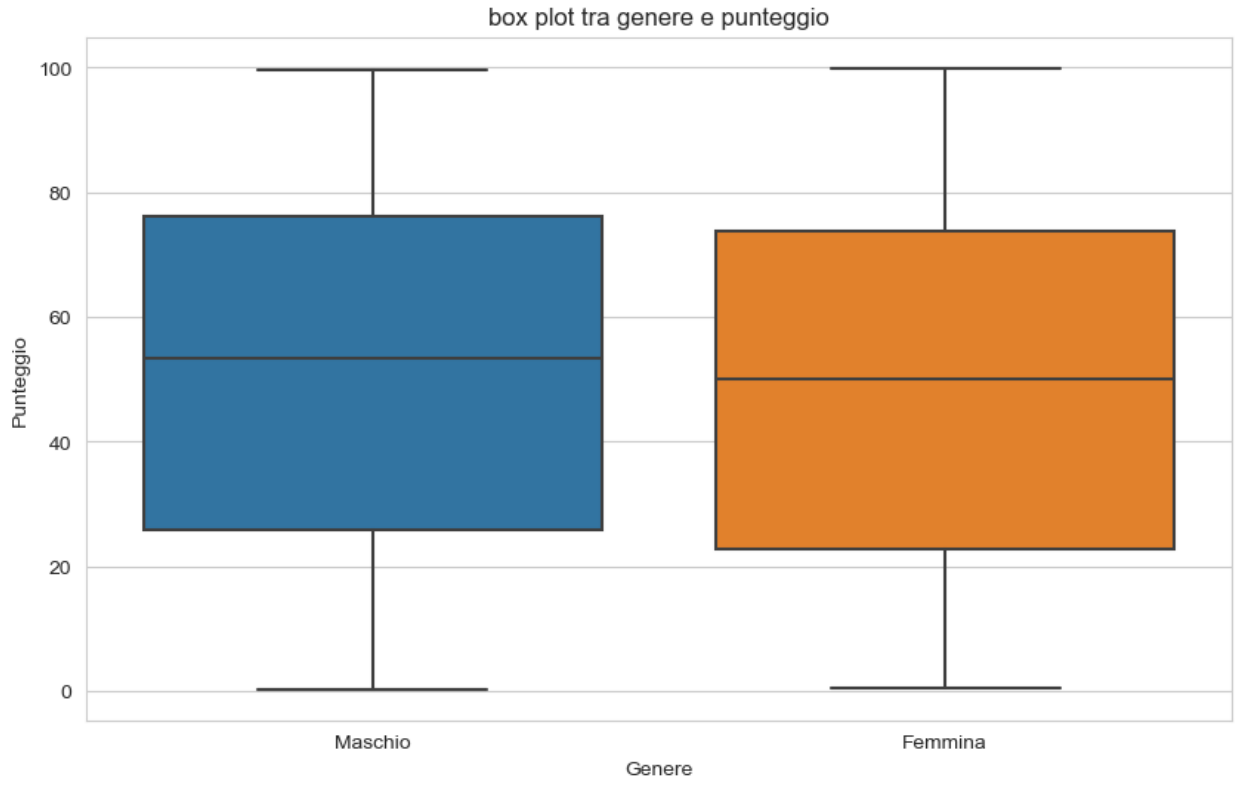

```
#visualizza una matrice di scatter plot tra le variabili numeriche
numeriche_features = df.select.dtypes(include=[np.number])
sns.pairplot(df[numeric.columns])
plt.title("matrice di scatter plot fra variabili numeriche")
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[28], line 2
      1 #visualizza una matrice di scatter plot tra le variabili numeriche
----> 2 numeriche_features = df.select.dtypes(include=[np.number])
      3 sns.pairplot(df[numeric.columns])
      4 plt.title("matrice di scatter plot fra variabili numeriche")

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/generic.py:5902, in NDFrame.__getattr__(self, name)
   5895 if (
   5896     name not in self._internal_names_set
   5897     and name not in self._metadata
   5898     and name not in self._accessors
   5899     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5900 ):
   5901     return self[name]
-> 5902 return object.__getattribute__(self, name)
```

```
AttributeError: 'DataFrame' object has no attribute 'select'
```

tramite box plot

```
#visualizza una box plot per una variabile numerica rispetto a un
plt.figure(figsize=(10,6))
sns.boxplot(x='Genere',y='Punteggio', data=df)
plt.title('box plot tra genere e punteggio')
plt.show()
```



I grafici intereattivi

```
#visualizza un grafico a dispersione interattivo utilizzando platly
import plotly.express as px
fig = px.scatter(df, x='Età', y='Reddito', color='Genere',
size='Punteggio')
fig.update_layout(title='Grafico a dispersione interattivo')
fig.show()
```

```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"hovertemplate":"Genere=Maschio<br>Età=%{x}<br>Reddito=%
{y}<br>Punteggio=%{marker.size}<extra></
extra>","legendgroup":"Maschio","marker":{"color":"#636efa","size":
```

[85.12069140487687,49.51465270139743,48.0586577326648,82.46809659251491,34.780920790219305,67.80161525590636,87.8629986355158,79.74260216069281,65.84518346584255,85.05817290942419,68.01407717603004,61.8611378215184 3,75.27166395576413,88.08707591989284,82.58167505647627,12.886986747344519,33.51188542591419,16.07598960483082,83.21341779577419,28.703813317491278,63.18135270166684,63.40057030996113,77.98453951511436,10.698063882695841,93.20281055100175,93.72284872364511,68.78857223008133,30.096356694681624,70.81720886452815,6.73506014687717,58.217046017627325,62.091551776747714,97.3488969177397,75.82631959290225,48.864319040466576,68.32953766065052,44.590270637674834,27.362666628166586,99.71245001577111,42.6181302235973,45.13870243296755,79.48095487499293,8.238104561584269,15.105789178090024,94.36159201675989,39.75943979627943,89.40992036791344,21.31047040250833,65.1666825375885,36.852634372376116,96.81934279147217,18.552551570312115,77.65968527917417,13.124487768238758,92.08478478156869,79.65372909761763,11.730818896239448,12.518579220255043,6.420893707513409,26.899340443509768,79.75591006371882,45.52201490818012,1.1620539908100636,7.244688779455888,39.24935564066848,86.01223971892045,77.98509888933498,24.204501586252746,61.81557343181191,70.2464970544104,55.96486834869141,0.9770847419183659,32.64613082439919,51.771164338471145,8.786649914483336,35.06269312091808,39.69232762015961,13.271575404306034,68.94649691372658,80.05866991090829,20.015024424481 01,93.62122462436898,54.12963353010704,70.90605194509165,80.17280830697918,33.94501925428061,81.48251137465122,64.35776951965249,73.15895217553319,8.162998203058958,6.03520839905638,21.92139873580443,97.58652558191314,33.68957917711069,18.211791568869927,78.96985071424791,65.87077755008761,99.63339160567419,97.4793162146783,19.954245092914512,25.76828885112137,74.2706521199981,42.54933344480754,37.10387629846027,98.76495637360578,4.010919141248248,86.70314961224486,57.86754085723934,43.86154191895905,72.52576604151365,48.666894142470284,87.342323808166606,90.07018640112585,59.23503285933622,62.296658356348274,63.156022009252055,73.3113022415281,90.9032520665641,17.968310887024195,23.754332492387764,24.723107440317705,87.07499012725114,44.53052550026655,51.48173539296902,16.352387258502276,39.10815366517607,25.813343270112753,32.51900642246949,77.34731256866006,13.087366071834982,96.98210450785447,45.37895413836392,23.605046334646396,7.349674733001265,24.871427258762992,61.714498660403,16.704190790094987,73.64020150656405,66.380452762118051,47.46308757498139,80.56701529500532,11.191961939772789,26.974961151698963,53.1169528001032,93.66056922949615,3.9343540668509647,93.38750175279048,50.723480868839246,4.15728590503921,14.834320096208886,98.66301229588468,63.91199378155004,86.79182945200222,45.47398556338591,48.88465802569202,67.34324333249174,96.99120461072704,44.375021930457116,69.26259522261645,49.71747640084273,61.78472402012139,86.89049844787182,93.09486955069778,65.12329770414921,65.7845304603036,4.821203886055603,40.690796072287064,78.23854840582734,73.84492092513266,40.4140321651033,32.70331615632387,76.22851347766158,43.5583314812933,81.78342161750216,12.020905537060845,0.575866049812157,32.45858299002703,44.86936226225306,23.754413087432745,37.325179162934006,22.726962726315147,60.34485933813779,66.82127985550163,46.34940438005724,37.978578017922985,86.33336495718252,51.908178513621806,47.9181877629731,2.56420658064356

88,39.88227808969005,58.017236922733225,60.79050927934446,76.48832615421283,81.29857387160935,95.55236959549347,19.57779856835461,89.8030548949987,92.70345447963066,93.44360268055144,35.16226872396344,48.58717591863256,25.677656772050796,30.728996492628657,80.30258978326634,61.03337833109184,71.61506742231936,12.188609333694755,68.11178539649828,18.143834769562694,52.516338366706016,70.90462617224327,10.687692309474384,80.59925497751581,55.022654215626176,4.341253285481916,63.31513755940223,60.16118201463312,81.91888594267903,41.10284698067047,83.98613028667204,35.34213793027181,23.687055897448207,78.05255152438619,82.26143185136785,66.75498992914704,9.553531383466819,62.3859324676573,58.66084631754741,21.673980009077564,9.571455552016172,2.3638586046336396,64.1971500443266,39.09059959055147,49.67668588591694,13.643975255673313,69.51445540475937,99.12559937021884,57.83869170064091,27.416066655980586,8.565824959643054,19.18673247187429,32.33715621155208,22.665640222698 8,35.49963058319373,80.03565078223616,65.08773690471942,53.29577865513353,32.43337531153756,66.18391898493786,55.77834173700249,46.52056122837182,6.0142342600785215,56.22968158779098,95.76252845053801,69.00048908206658,20.093368865905646,53.582768447308126,9.667644964671961,34.757152423898205,66.4911724487218,79.54499607897291,92.71778194794527,23.464208185545353,39.931591580930814,99.24835021583752,84.20332943764213,78.20280876703859,26.311256991084285,8.764274676383854,55.38022407123763,30.552431065802356,60.05943333606173,91.93919732144174,49.696348290583025,99.21580148233127,85.14249577450957,20.85105143573358,11.636639788547608,81.74497084911916,87.79743204450695,30.46791390980408,8.091928305125219,40.29801787651615,69.49510889024617,97.56102008599149,82.24805636343609,13.252467274184843,86.20144822726226,60.625293775097745,76.48098009029954,14.63739921622491,6.817230938549357,13.516628915482276,50.61418628402441,21.60381825095996,65.02010923981534,78.07615860684398,5.816379550547579,99.48663171697672,69.50352288585532,14.224936868719617,12.138493942242889,30.327514756759555,10.10458110106467,50.94221281497897,62.389583629163845,52.70414639116516,13.071037674019948,88.66042152404667,44.97846488213225,82.75378955668742,1.1031264428647214,1.9192276823056686,25.98132087450342,76.0289823122135,1.2120774643893273,24.12014575995415,95.95766443210334,48.785404392636245,10.973619867750351,54.79594889736929,84.4357082739616,32.467592689006,73.73570773877996,47.601812936876655,37.58882851457693,45.94467684059866,78.50165438370568,95.53346865039283,46.147746462473414,63.72014768082804,65.93539222057827,89.51177407065364,61.39335847600618,51.84080219452041,15.016900081704165,73.74337687214856,68.02277792083532,71.63233880626822,7.208433599867337,7.12567277652586,95.65013982994802,29.65355812876552,34.97032297458495,17.410933533408436,66.03027197013381,2.094496103091281,96.78600348450304,29.544477747742025,62.46635688542638,38.19396396528959,12.138641815940643,61.50129680388666,77.4633780637788,53.03021332491583 5,79.87141916545745,29.282203670096475,97.99703293588506,58.24226582072465,74.8073179429532,81.17697880901376,65.64786073315027,12.809574636620003,22.461561184385193,37.21670238375133,43.2076882944196,43.940498969445215,94.30758377447076,19.747048985993732,86.1369996363436,84.6514338690278,25.224102017336158,75.50419287462783,46.053949727414,84.19985528900615,72.84906771616131,65.61618376810715,17.74287694

6970686,54.502691889693835,98.46697395245081,72.59799099852955,21.3511
3663106719,50.5852692951829,84.07030280112,73.28015447662506,54.223720
549184826,68.88853020500906,63.629135397924806,16.0071634954573,46.155
74744197899,24.667886452162303,72.64617151130048,9.917809958721591,55.
50849493022862,73.30712960296748,61.5985450260179,35.538456868573945,0
.5229613542912737,3.531135494023907,74.57337827312321,20.2480560165237
4,95.80734801193802,36.79407513020845,87.66507618282006,99.63343376272
474,36.83095305068483,72.2070938202737,39.15256981778293,41.2621840848
62376,0.32182636042786816,35.54930104413574,79.41973312770843,58.82022
669483905,64.23255259052274,56.14845879595237,60.34876665964548,67.646
79372263622,80.49889996807697,49.82556822634536,7.705827913433117,33.4
2383186960608,78.48969771114676,78.86149646724064,44.01989979554906,32
.81927536093116,43.40193661738753,22.061195271121125,73.56631142391899
,63.617736470823274,78.56516180802319,38.3832951933044,57.187227122718
65,58.77693608621982,18.447625313363936,2.4191763891355245,83.16970436
757735,27.30708099728124,94.06792451153404,42.96568127389351,87.273025
04846018,18.610141784627732,45.81868867378927,72.79393069737652,49.646
115235529976,76.55128989911097,15.890816766935355,61.02251494776413,75
.13750860290457,65.69551562671396,6.895801635642118,5.705472115125431,
28.218707469320016,77.67105569551812,6.536615756438524,3.3613600183282
633,6.265320345535452,13.924537111759516,53.24206822752171,41.10956026
007065,34.73433262588427,89.98333456872724,56.016818346184195,5.225787
928602377,41.87933190741614,26.015779096205115,73.08209649807299,65.41
746014740745,56.53302545716099,60.85272726964666,11.409578488793336,22
.53169299512351,25.0966661340298,52.33909113743665,72.28143032084837,6
.766836190000158,54.353821734264606,8.172534574677826,45.8300641528656
46,94.56981470002823,66.90223373291083,41.17655415845085,65.0973467150
9541,6.227310588188562,45.923987759695805,5.195657755426842,20.1363782
1359744,25.862083488803876,16.47063534322959,33.02150648714976,75.6751
505688635,87.78300781520409,87.95818549515292,87.05784250460069,45.123
93441809881,6.5204591260492055,90.92202058615601,49.781250803429025,65
.6780104724294,38.04200075075623,77.56118509253275,96.44766539008116,2
0.376645912768907,79.28544190338394,63.45824175503038,79.7914159615894
1,91.50900684070432,53.302886724614,69.58991181688025,79.3261350438254
5,31.67616770675267,85.7179256998753,14.071152804875842,20.20156684024
8997,89.39897100078704,65.42925523249619,15.210428322227287,44.0323418
36834895,80.36035354415351,41.77609792750997,98.41103190364979,42.7489
6879206152,55.975568363838256,71.93536223691585,89.02580530178021],"si
zemode":"area","sizeref":0.24985343144266664,"symbol":"circle"},"mode"
:"markers","name":"Maschio","orientation":"v","showlegend":true,"type"
:"scatter","x":
[56,69,46,60,25,38,40,28,28,41,39,19,41,47,38,50,29,61,66,45,64,61,20,
54,35,42,31,26,43,19,37,64,25,34,21,23,59,21,46,35,43,51,31,31,57,62,4
3,62,46,32,42,24,41,28,34,52,50,22,24,25,29,50,65,40,41,39,44,54,18,22
,43,31,56,44,26,32,59,30,49,56,66,40,62,32,53,30,49,59,23,45,45,47,28,
45,42,56,50,58,20,23,54,61,41,49,49,41,58,69,66,66,69,29,56,66,34,66,1
9,40,54,49,61,43,49,23,28,34,41,69,51,23,39,28,65,33,20,37,36,36,37,49
,69,56,35,18,28,45,40,47,59,52,24,69,66,19,65,52,36,20,50,66,25,53,37,
46,63,19,33,53,21,31,38,25,20,34,50,29,68,55,55,62,68,44,44,38,47,50,4

5,64,50,36,21,66,34,61,47,63,58,41,66,63,52,50,38,49,20,35,48,57,63,41
,67,49,44,59,19,43,57,50,46,59,52,67,42,30,53,62,37,25,29,68,40,32,49,
39,42,39,23,50,25,61,56,21,23,62,64,33,30,67,34,36,43,54,43,40,29,18,1
8,64,65,42,57,62,18,33,22,20,46,64,47,69,22,43,65,38,56,53,50,54,40,22
,48,27,36,18,62,33,41,33,45,49,29,52,50,18,60,46,48,19,68,52,43,25,46,
43,51,67,62,46,53,42,54,52,65,53,35,41,40,29,66,30,40,52,66,37,65,42,3
0,36,53,48,69,36,64,22,29,42,69,53,56,62,59,31,52,40,35,29,61,55,24,30
,57,26,67,44,22,55,36,25,62,18,39,69,34,62,21,53,23,36,44,20,62,67,45,
37,45,25,56,18,20,30,55,61,62,49,62,64,54,52,37,35,58,66,32,33,40,28,2
7,25,29,41,45,25,45,67,45,45,58,34,26,37,45,65,30,52,68,64,42,19,67,62
,50,18,49,66,38,43,39,66,52,18,45,34,22,48,40,54,54,36,38,31,63,68,18,
66,51,45,48,56,43,20,67,29,64,40,54,38,19,28,68,56,55,51,47,32,44,68,5
5,41,59,34,46,21,27,41,22,23,19,30,65,68,40,33,28,21,57,42,20,49,20,44
,67,68,36,22,39,23,22,29,63,51,66,31,44,43,64,64,47,65,34,43,53,52,69,
64,39,31,43,41,43,31,40,64,35,55],"xaxis":"x","y":
[52915.76452366004,44702.505607839674,55077.25765192536,52526.91464404
421,69763.9630341194,34901.86152080538,48228.97209513382,18172.1765148
7108,40882.670193583515,69454.91819841664,58201.00352302069,46416.0185
1245458,44497.63382277969,36163.84723252246,68257.37781238244,72819.74
077883677,64974.66347221941,56055.95135504914,36444.47208904808,67815.
19082692346,54249.31807319213,58800.40702212912,63069.45946185866,2981
0.304796298773,52693.72672909397,57337.80695221495,59520.82202842655,5
6147.279853927685,46381.13518072186,60088.60551804097,78498.2290234282
3,35382.06042994038,17409.95680978946,54979.70495853522,57696.58697264
124,61082.157213312545,59230.5122601958,35968.41946933107,66289.731742
62549,41960.54832328928,62120.86697421718,77572.75515192683,49708.7059
1168401,33982.9561763121,75659.20058872888,51050.78244669772,64627.964
44673645,52843.724248981154,9451.515605011446,60168.12979296361,57668.
03899287471,70604.87817744436,64293.118208044085,51131.50458352389,463
09.062678050774,82564.1407624672,51848.072101767284,58272.280640607016
,38862.94121452927,40823.46363642072,49444.44804427382,39613.685301892
16,28905.23804492249,48753.41641056958,27429.194389534965,38717.653916
14125,54787.61765628771,47598.00806650039,45214.19096500073,38059.6121
47947664,66140.10707897898,50319.67473007913,78517.86028629511,49090.0
8779278069,39373.898496343056,27294.28411080759,54006.89976517225,5763
0.87534844901,63425.57471114173,42754.08422474841,52201.89517978324,39
212.7002113637,35192.30709273224,48031.145493615426,40249.961287648104
,52529.82007950092,56629.109743541136,53498.24196895253,54306.86247802
736,40813.43938039343,55422.55287809153,46878.2471409119,48071.9373284
1604,21772.264317535413,41769.12638388558,51392.673517116724,52397.845
87996101,56407.91542807527,40306.59035234295,32095.447523652798,70886.
97950096492,47985.368877941786,62162.124097396736,23772.02040142729,69
565.10364317229,25062.620179464884,65488.19306512486,66900.57705241375
,33635.50354677203,43837.78568068123,33414.42997271096,46776.184545668
97,45379.48573721388,70935.25716184934,24748.427116685747,37911.949002
55717,64472.77449646186,41113.036112706584,49603.91364561609,54202.417
3822941,58194.26372005549,50096.33404716407,43454.21164210147,48355.85
2941112535,46118.059059993466,73979.70755757076,45567.795229562595,449
92.70765365985,67596.86956379961,55544.63288297141,48390.467839936566,

56715.75388715416,26436.85771897251,33098.48796635292,57512.667124166685,22984.133987708916,38183.24372137642,42918.64133454309,20310.502844363185,61218.65384176301,53588.702961667244,45709.936671798336,66514.39452497856,31556.756083213746,57450.48834854783,43020.3431482992,27455.4450740727,57015.3975764815,66286.7948740902,51469.747302063435,54620.75809256153,59323.57834784076,44455.84495934549,55656.505908532796,66890.75376158251,70616.57139756226,17964.886790319568,53348.5837119374,57816.836425690075,47039.923695390105,47730.90074519732,47076.375794709784,5894.170480035798,64212.89693184287,56105.78430686802,29305.727528166044,62935.902085741385,57696.28130145939,57742.67430672437,56478.83817010911,62006.14287841524,60624.55770801596,66052.25357474641,49602.1811112641,36771.88023158423,38826.46032790326,39872.325875384275,53199.405610594076,50018.082130433366,37743.67053397714,59888.68502648608,25886.602034465388,38559.12832150679,35901.45348336703,62442.12262897623,47092.60791840795,46032.27738034084,19942.064533898698,59531.27045313839,54160.654069372875,70409.87893757976,4707.317662687623,52757.743171305425,77007.66772266281,53144.891229119385,62106.83894955966,57582.0523874994,91394.90058366617,42365.54561438752,49616.38531712008,39579.303033664124,43860.76342830344,52285.32308050583,37663.70101426699,49860.495182920786,45081.579252306234,36993.0467643365,45444.109568350003,29811.936147901775,37711.13210417754,42856.68689683011,63115.837710690092,55856.972748933345,48451.67370601256,53980.435240994426,41258.62013582627,47985.81256720544,71341.2220414991,52959.73561108393,40735.22156848475,59236.56724656712,68058.26655753225,47908.305864785005,50007.91938284896,28342.177757996873,15557.28570294143,41741.94725762625,42377.897035290334,54901.1714690348,54507.11539416242,59333.10716065103,32917.50323328846,33922.14150439767,62268.344870375426,36469.226684637346,36955.053780879585,59575.87888412575,47558.106408915584,50613.78763093582,61112.36518130189,46571.00126989972,35084.75938087232,11564.995118070343,47134.58356099747,46056.637561875985,66714.82542003308,60730.719731618665,60772.785885619014,58877.212006120666,44689.38933588464,51527.8444289298,73235.30227242781,31413.393865191487,27987.1210137134,50763.31552784027,52600.124965556126,53659.29796862751,46651.966180477924,37572.54462829631,48451.17859776429,25352.16783268749,47362.18393351893,74922.05248592027,50313.29732443942,31097.528522411296,45234.27354865649,73708.1102985256,46489.70211339528,64810.01774883653,45202.28715117734,52276.3698837311,25885.08459322912,80354.0932884446,50101.99766839996,47149.41447497758,44638.31865784359,47294.12477356388,16822.0713300574,73001.50551944488,46000.21506148378,56932.58999122914,80364.64434930279,29552.389931107842,40070.26733305086,50287.21666123628,57318.08427306611,77065.22164680115,47136.44312409052,35653.4543477298 7,57086.09416820566,49989.37108135031,59067.7298953305,43218.645007503794,46294.61448881005,47281.637401728454,69819.56048409519,22001.900729590758,39730.55261359994,61859.38398744153,63388.96007448964,50214.096017546515,35690.90870905033,43894.455306098294,60294.77685435051,79646.60876773228,30045.295613915154,47163.1941383445,48086.763194336985,72667.3239537102,28232.364649226616,63303.31085145951,52984.22220194742,68536.73287733231,33934.19480526427,60203.24158175674,42437.302534884926,48777.15990347531,39864.37881257582,50512.28626232037,33691.310917

05796,33712.62306672222,32768.08947262016,39836.530214965176,67911.641
0457106,35282.517032904776,43033.93606535919,61750.85763923893,46226.9
1670933257,71335.54792547761,59756.75138804209,27453.79547360273,65779
.21426584378,35029.090590283806,53753.00309377542,96648.7783400804,591
00.84588704274,47252.05071919418,63314.831388107545,53451.06138787103,
57466.14633637091,59988.86378081692,62584.10750704419,33497.6865054245
7,56592.51845930339,49916.059717564305,33624.486857910284,62736.531661
147164,48926.011222129,27231.89379978786,63355.74745279679,58628.07524
159849,50746.6053077144,50105.94737942264,40095.18290789829,56314.5941
6584085,17699.856201931336,66457.29067630543,42817.438159873935,60402.
1836787099,44119.81040121306,65899.04576069933,59255.089243650466,6025
3.53984800716,53918.75796850784,44460.84291130695,52150.82701013377,23
356.471506525202,34559.42737726893,66690.32562484888,33252.13340077011
,68537.17430010409,41775.69055264098,52400.273387791814,57526.74104251
3175,66760.9822385369,71727.4884483939,30109.284708769424,43798.022583
15086,53904.20759874329,35543.61610684084,38622.569979498956,36136.502
835975436,63352.9761668718,65528.74002085629,22307.18206642859,36057.3
3709711385,36172.1028031203,53109.00994098416,39173.9366999412,52652.3
1308209097,45925.1733451092,75101.78135447805,30506.281446086035,62281
.66465632291,73068.97877618829,33161.82790633379,65264.922002652536,55
108.83824600745,30102.918075333757,65709.76559932172,67543.8498864166,
46559.13883897958,49347.84575279206,49067.134745010335,66865.300398057
54,44988.84153530089,34705.034919618265,53379.61883061413,44457.096067
499624,62390.707571622595,25901.34348891981,76243.75856308131,30616.05
957341572,42455.37844172985,46483.88299525517,39641.88421525611,36257.
12126778136,37522.66687654333,60230.7809856893,72308.6923300389,53591.
067508030654,57495.27665454401,51139.32901874131,57233.73429020986,630
37.340965839096,63309.36255900113,50569.067531714034,46860.29270630958
,54557.27893407314,60823.43757689215,60393.77934547416,40912.087688616
6,75790.66859310945,79887.73029752178,44193.04025315728,49318.83173101
643,50380.82630813615,39654.08338903739,71654.48005203222,31152.396585
811155,45815.55604502287,61855.57844077567,45030.374252250156,40816.44
898238374,94236.41637963089,68670.1937110518,29733.88450687968,57227.9
9092577737,58208.97196066951,46175.66309669453,48118.186099956496,5491
8.19286239036,45297.054707414034,63827.02252965878,56301.417356777616,
59102.758898235064,33036.673470111265,57104.481152171815,36104.2830502
7692,58332.689695133726,36219.70232770858,63776.81665913621,50824.0116
6499773,39356.30217876241,31122.42633986998,46744.74424363444,56494.39
7380410446,41528.389937435866,26210.250316430625,78064.47763057673,369
75.6090218545,58019.43629183653,54917.31759264628,51285.307421050384,3
6118.63060910619,36569.80516455464,43878.48741259669,35062.76907814656
4,46480.69145063879,28271.128780496838,49246.93074437383,89311.8963654
4962,71962.65858128657,76379.307262927,55581.58208910294,49203.1902047
3682,68353.53323627941,66447.03850716326,40614.84068755682,30211.29208
1018488,49047.13138076967,68561.57202026209,43140.47517369815,50870.34
1022125926,53169.25538419209,42621.46328740692,59295.67168949205,32155
.00011364458,47232.11963516226,40928.83003192681,16916.509143513962,63
455.99295330698,32934.714943627754,49902.18027155813,58981.91584342865
,38987.39301832256,51229.933346203405,71837.3348322015,60567.598097370

85,61835.850310632784,51257.39941894041,71050.54955187786,60468.549088
08541,42807.60548901135,54448.63317711143,56947.77192901377,52959.3839
844681,52859.53386897536,55968.90574920213,77139.59524599665,38228.571
37808445,50234.27500924405,36383.65729779619,72789.25885462653],"yaxis
":"y"},{"hovertemplate":"Genere=Femmina
Età=%{x}
Reddito=%
{y}
Punteggio=%{marker.size}<extra></
extra>","legendgroup":"Femmina","marker":{"color":"#EF553B","size":
[59.24077846595176,56.5731963995791,26.70282701694213,86.7294200959802
3,70.83629767150347,83.7013328363672,69.74714616692836,15.860510529307
37,87.1843527745232,2.9247283034559146,74.35082562916078,81.7967024119
0622,50.746773376083624,0.638587171683358,61.69269183757422,98.1186178
0274235,25.98035810641598,53.99853797158555,76.1027902502028,54.126657
86761103,96.29920038589947,34.187216603868606,63.26218931339481,10.250
972799067926,6.78370591051719,34.588305695295674,4.5742033812512854,87
.15368061523763,96.88778552856915,74.96518317429248,13.008624013063198
,2.458691645880151,2.2123551528997254,32.36102191495412,77.04074178077
931,16.362382119166917,69.36822257814886,22.07696127887604,68.04993020
74713,65.45112142811352,27.32595269982093,95.08635622504103,43.2334801
0426019,41.97273169261223,63.85259476640856,27.421520234867003,98.3977
6479598282,40.9334006315043,22.995460589108617,3.113408288451136,86.43
58249839637,47.32099066915572,86.86231679556109,77.0921844644052,84.47
832281168877,76.10239909429988,62.62203216314154,3.2526179491251916,61
.665031452073805,48.152235151255006,68.55652872289714,43.0305894899462
66,20.052472670033673,49.15945467414369,58.19714019143988,31.036195892
357377,47.99388347171235,60.00205481193037,29.166257870930323,69.49818
861265426,3.9618825348357145,48.0506947257829,10.493017841817487,98.66
625932671465,14.249554290151323,49.888815345135605,3.3203108791366565,
7.857849715502074,56.75408482615582,16.748258225906977,10.456784033440
025,63.64302495436367,70.64757264869012,3.1586144825642037,5.197128365
1472415,87.09691237460856,71.40869321324278,8.011484638467515,89.48166
560605276,54.75923761537362,81.72977699624937,45.23182845183001,52.640
266093611324,24.710323401014644,15.954468011318834,87.17835665922017,4
9.8195716453139,55.53635509376312,71.92017782722638,22.845474133129873
,65.03256863469369,68.02282424312914,7.219840897917584,3.0652502205806
065,46.26229567393163,86.82725054083805,72.7169069766308,34.5934992546
9629,42.17209268734554,27.682779723505156,91.2363345616691,21.06621890
0566657,13.15676851272598,71.58249646820886,97.13950940416396,18.09769
5270948975,85.4385093369579,49.227785644803475,35.92333693997657,59.29
508514349274,96.94123223352875,65.67366645412922,16.975790508754073,51
.977394855601766,33.70031764312863,82.88833658826094,43.08875236618059
,70.67772168854458,16.761921628831765,3.667142693354297,84.41704489691
972,58.535436439675216,86.82712805132586,20.584121003676824,5.70868560
8931486,12.21099140100268,45.21990282834353,31.61561049767505,96.51186
964360456,0.49399809344096157,95.18117854232389,51.55960285792502,66.6
8642575461044,13.96512547563129,2.9973589872677953,30.79299415911909,7
0.46807627366466,20.185345212294838,9.39007157894184,67.26021182251263
,86.81422543775054,17.71497894473859,83.81152896481102,94.461421947402
33,68.32480282896458,57.060974663882966,3.038705969683686,68.952675103
15611,67.65133857772466,21.567515239771716,65.88854702326572,39.386440

565415846,10.659303030799073,99.94137257706666,97.71741842213173,87.07534503372591,56.70162609866678,87.8515561301348,66.75933855635847,80.78459419412586,79.78136488586345,54.448909796084955,36.646153487139784,39.61726916092554,69.54672066898621,38.85581012703457,7.319592376844364,61.94903460096145,34.12478276261564,38.019561878585705,53.360254672687844,71.81230762264944,1.823258267636163,0.756287497814212,64.7474714154603,24.34822965762993,6.026739028956085,10.142082844921651,28.487290196192284,53.91612766752545,31.130769915600364,27.262400073353376,41.354910152535155,18.11493495346803,56.73122196289035,25.656278370136377,96.29268752375798,48.35456467199247,95.14033422066571,88.42064633484078,22.807977194893247,21.204483990822666,61.09809888509917,90.00231233980112,27.48060355481361,42.37382536487628,45.17676787372291,16.801420774486342,73.68737449441821,86.27970775432286,60.70940360887808,54.669741271407005,23.194709611597542,59.4476335186173,98.77855202148874,40.431876810429216,42.819961492561475,71.75977656581762,69.24361513243696,12.839428945412523,10.410964938416633,72.43388171287411,7.9419368988986605,89.41908746641182,6.942384274609337,51.90597909082704,6.76125635922149,23.3712081910433,54.00119146045106,88.00790875614582,33.300191305819794,66.94869551981901,99.41393612211675,73.06505102392461,17.530294235765965,45.037093625818514,75.61633288952461,15.241601333041743,92.70009679242953,53.99571302805596,52.09579772720422,62.35856880027597,8.912443211601662,75.52704204968079,12.771348363523483,82.60676302525283,70.87446977429764,3.6160380524762115,30.31283559453286,36.013640747715191,93.69578230310628,39.698151707428345,44.72025405048357,51.567942769649775,93.05952146439162,38.06232934765801,86.80566901074504,80.59254001176211,79.00304363070614,17.35245149865087,34.609972653778556,64.0972077345332,92.27571905133944,48.70619188189359,17.483862726041256,50.25660764634424,39.86630277723131,36.753442229153144,2.58119069494869685,96.31151115891738,54.952953589501696,96.582216115322,43.2497878330132,31.181613308525957,43.95116886329101,10.566468426174392,64.08263146902063,61.95879548978296,15.20248531286481,6.134962711066816,45.98004237693202,5.778056099636664,98.36789252634136,23.918010421592562,69.2161341694132,6.229179813006269,99.66968538529768,81.39702695060188,61.52194382310109,30.625362078661166,42.60833838903715,19.462251182076507,36.77593508772896,41.4129752528927,73.3614380561402,76.93048936413398,41.615399817976154,48.13441516353236,13.710970572899905,53.531013145259166,21.52018720172375,97.58737549100537,80.15371118829896,45.43773345599609,9.808258078424458,48.82411370217782,15.004866469398426,39.44762785700353,89.20846889402344,78.6903386255295,31.540689574801274,68.81347060865421,43.760312405539196,25.467062204605117,84.08715794259226,3.84263648962324997,90.17619921354235,63.66696821672073,6.665204056068996,51.22219208126745,4.167290060238427,8.479201604580322,1.210847523488745,73.75083594590069,35.32514085020642,77.46535311811819,66.13706110704143,18.519556786746794,9.839564971339732,76.43726622217359,26.5046428325004,8.217166747642734,76.92231478381336,20.56872621955551,64.39042535411306,4.195122343163627,96.84887775943838,60.18815896363221,33.82675059944984,92.80836403260497,61.29395824945255,24.069271183250496,12.150137630091805,88.69249002092695,64.5810812655206,28.590678923916013,81.59469389686173,91.89265336395545,77.64474464997247,93.73880664971189,

4.317373579701167,16.481481572943714,13.172877345456191,81.77853298569451,59.03476904587729,50.83605469162799,29.754845131882057,56.50219928590161,87.33229158608077,76.1121537043158,0.9331619827093296,99.18099513493634,40.14943158612883,80.00709686991596,20.403563462889885,18.802473461541613,78.37917844853972,55.42265141320166,76.09907600280938,32.69316163380791,14.888805033244468,30.560421549021488,44.861063095454256,88.61957804650746,59.304433418678414,69.56181464968068,61.95893374598407,9.299063982313216,48.09728898861238,6.485359448296835,57.99837874281313,56.06600944161709,26.982072012821433,82.5049410180062,5.85509292439792,70.76809347409157,51.72690564416094,14.745252670628394,8.860042967961723,59.82252940637388,99.83475113929497,93.31133313435576,64.25651995351916,42.1248053294072,11.833619092338733,40.99048896962883,83.98022856356172,36.22354416522884,33.451128874193216,2.6196708532064794,51.80787662388154,29.872557358877305,25.929675886577375,84.19335667918087,80.26433097957104,48.29688720853552,13.347997241594378,8.060151375712465,43.68507026151796,72.95082286523967,13.535408227768553,95.6614621083458,26.170568373590598,24.697879907199983,90.62545805210367,24.95461998495061,27.1949726128631,75.93982624179637,44.973984245054,48.75711936733843,90.64374533444109,2.18233967754895,66.37896861755891,96.33944342135504,93.68224620330918,98.12970904972467,25.65300624442478,19.809763275974014,46.39324866883362,97.2005329631537,34.950637208681876,15.124682873493455,85.06160578411378,56.12227882825249,11.476887186921747,86.0139693349435,70.78350972297687,48.469628713004695,16.577454016327575,84.99753714806619,46.22955610006072,54.543186940690546,51.25026487703317,80.64036172328395,78.6278199384033,51.93858595039374,20.488127415071254,23.87962113815393,98.49896561408751,77.20124763099903,2.716742122555982,46.39313768123328,53.870179795271426,10.547369995663058,82.21031606002255,52.33296311947413,28.713795808457554,57.75933658811575,39.59704726624549,15.7954822835348,90.61432547657664,27.690448737780894,98.3521472715222,18.422483923802613,61.529802506980204,8.34640849893833,88.24163684073388,50.52067735766382,96.71266152591704,66.79199988036271,63.467127325394145,16.5954895295785,88.1927758304644,16.223340578247935,1.260751997575904,52.739962212267635],"sizemode":"area","sizeref":0.24985343144266664,"symbol":"circle"},"mode":"markers","name":"Femmina","orientation":"v","showlegend":true,"type":"scatter","x":[32,56,36,53,57,41,20,61,55,19,39,42,44,59,33,32,68,69,68,24,38,26,56,21,67,45,24,61,64,52,31,53,67,57,19,61,27,53,48,65,32,25,40,38,33,35,64,41,42,58,62,18,26,18,61,25,41,68,25,52,59,56,58,45,26,51,54,52,61,57,52,18,52,64,31,20,32,43,68,69,49,21,47,54,56,60,46,24,68,39,45,19,62,61,61,37,18,44,69,30,56,25,44,26,50,68,59,32,19,20,54,66,19,45,50,18,36,19,49,21,55,22,50,26,23,33,46,53,43,20,24,58,50,57,57,42,67,48,33,43,65,18,29,22,54,49,26,58,65,33,37,41,41,69,28,55,57,52,65,42,52,42,35,35,52,58,50,50,65,37,24,65,39,39,63,47,25,51,22,65,52,45,46,23,52,54,46,48,69,40,50,42,59,20,64,39,40,19,34,26,60,65,56,43,41,24,18,63,33,31,45,51,19,40,68,39,66,69,59,32,60,54,61,22,49,69,47,52,57,69,59,47,36,45,26,51,49,56,39,46,29,43,33,68,54,39,31,45,22,66,63,29,33,43,47,27,53,51,69,49,22,21,19,66,44,37,41,67,50,68,60,54,29,20,50,57,27,61,30,29,63,67,40,34,27,43,68,58,24,21,28,38,53,27,26,41,66,52,66,56,49,49,54,42,58,47,34,39,66,29,58,36,29,26,24,45,31,33,52,38,40,33,56,48,22,46,6

0,28,64,26,27,34,63,59,19,46,54,65,24,42,48,61,56,27,43,36,56,30,58,45
,66,42,50,23,38,33,38,28,53,36,64,31,48,18,20,29,49,33,55,53,43,25,54,
53,44,50,30,46,63,23,35,22,27,47,22,35,64,28,42,44,30,50,51,58,38,65,2
3,22,55,20,27,27,34,26,18,62,30,21,57,49,25,51,18,61,22,47,47,34,65,32
,31,51,55,67,35,51,69,50,32,47,68,22,27,34,34,37,51,60,60,28,64,48,33,
25,21,46,49,38,35,45,59,38,18,58,43,43,62,26,39,60,18,25,66,32,45,40,3
1,19,62,68,24,20,64,63,60,62],"xaxis":"x","y":
[45568.97884791456,67098.17841693568,69756.72601009798,49656.979065516
77,35010.466515576336,42428.376181576736,62609.300397568295,44123.6277
6434267,74230.63534204163,45165.192961829904,43525.695309229275,49637.
06590413626,54865.389225824045,32314.40313260296,43030.7405405241,5301
7.39483809259,46116.42539443884,42876.44515066213,51895.693693196175,7
9083.93498587428,34995.03013091638,39833.82544212723,57708.61774488372
6,55259.45148910339,66645.49762934334,48010.493804261394,66606.2100393
046,48194.282540696026,62711.325157256324,41970.07721069116,48642.0007
6543759,52857.495196685995,60641.77725708559,43467.70444145235,46106.7
9983624124,55509.30970102011,46648.01026442222,44760.24805822704,45452
.303257485866,61999.12852913075,25755.34087345118,34194.76371280469,64
254.61387960364,48433.261721937924,47467.67415214984,67428.17453912286
,36089.70298714139,53575.53474102178,57516.41254909768,65015.691388507
12,40188.864753088645,22540.506547153145,47938.27228058462,74184.17386
982968,69723.7168037607,74599.46794057089,61131.91236607838,25970.5128
21542474,37351.30107025713,47361.712813514954,50654.03669558681,75425.
7656064613,40660.25957398322,52919.11182855864,30199.662331689415,4356
0.4666322411,61400.83945534404,51236.59629401755,28136.72750802604,453
61.863844252046,70106.75669034716,21872.41295531383,51725.39118721126,
60070.101146756984,53197.94942356534,38720.460023917716,22952.90485616
0825,26237.960854917652,26282.139469518,74183.30942383136,63452.589737
98298,45972.04028141251,36632.11661877069,17722.768980060428,46833.046
417584585,51152.77824438152,46627.16027287882,33644.01383835731,71163.
98569619405,48521.17801573608,50282.744344172716,60623.21634884836,642
97.04950869868,32844.10689019007,51628.39558664785,49501.550140333085,
34584.87288611063,68985.61761297051,37007.377276782485,64541.858007901
05,76629.66338914268,63787.31260415002,65008.734770644674,39940.696863
85033,46249.302281015705,54330.40443919785,53904.82758156831,61902.330
80828319,61694.907984040095,69654.63119179412,41567.47960132547,46861.
671893832085,74233.74235322347,31484.76692180117,37855.94720117531,563
60.915666961344,42892.409793936255,49783.215936825814,48674.5556119199
7,44449.83455176809,58413.791719118584,60454.316067661035,32091.124904
01417,52144.1464214556,75987.72499227637,83469.50219288949,59570.76652
161136,41859.89300398967,40687.28602623326,47477.84272122815,33908.854
613902906,81111.2400462496,36209.23112620365,12045.687158805056,79375.
21181330891,48410.774685437966,89665.15020525552,53806.094922418604,44
125.271488673134,54036.90411867661,44852.11586753884,49561.05779472532
6,49229.094146172705,23404.512976896778,68928.82251070987,36414.015487
060715,40193.504920267704,41065.08058934762,97066.22800489991,65840.85
213599078,49176.5899907557,54283.311122864856,59678.23378925085,58334.
066990051724,51343.71021882873,67006.55145300596,58903.35083421293,598
38.51164794641,52921.04292078162,49719.36865331154,44172.22320366671,6
6861.6907997902,38406.834982645865,35425.14898188071,40599.24163115052

,64296.8756806413,60876.43680271995,40377.77607165572,61314.36994720361,67833.70057211965,55271.72311312374,47553.99554084354,47832.219939267234,38113.70118500058,45380.5770554144,21595.779995693214,64063.55206439838,38462.86475319053,31411.12407719791,50898.989453994924,30367.69401321332,68584.19529409157,42625.45721551929,35396.80700443297,57145.36654034329,65903.1486124832,55886.23950768151,23463.861426663225,42138.674030981114,66815.46000986706,50003.10597377296,52327.860257884684,62376.47397001452,40128.25266387115,53938.419935028876,52903.848670513165,62763.46943251958,47939.42317400905,13417.742625853563,63893.23103362389,64480.96226886756,68541.96082709327,51329.86386442156,45258.90715959425,43247.16102722882,59018.10123393672,31689.31361128762,47783.29521773793,43201.2738646548,71787.01301706118,65586.67681130403,48863.5357019681,60057.20925861819,26693.618509759035,55646.14289146275,66881.52176622154,32158.82160699008,74640.09429883945,36490.68898028197,45070.72286376523,59047.81146623514,41838.289968287114,34967.18816327791,42301.79650292622,86189.23132419374,61769.06431847036,49711.09430398408,50336.9883921807,58206.78672762815,32287.807770367326,56577.13608214591,50294.249865409845,60092.918539502294,41395.97340654649,52471.41261854775,72347.95157903123,25986.440322852268,53460.51358578025,40754.57887940784,44372.054295140944,69224.65692505654,58365.36591087687,33328.13064594452,53697.571676480075,57473.32622745178,67102.235577819,34773.58712258197,37837.137331844926,31136.332116280755,56995.375616695024,48860.73464536216,37472.856475549655,81343.08082542161,52771.108683435865,70592.72712014796,28640.642713247824,43561.33369552007,58828.299046560795,26028.134721743787,52845.592514155935,56388.30809289554,40377.69636433535,60796.369132222724,30600.905631932543,72261.74042803947,55334.2002317849,45304.12954215269,31243.883990454236,63234.995934851133,42949.36778360748,53988.173444780725,43449.20395939107,49008.01078061266,81495.82683749773,44624.89771968167,40286.87291547261,61162.87957249905,40259.40442517861,71294.04506770456,40993.64651391942,65112.705347553834,20448.442734322467,31832.42106078828,60961.46178599787,51588.41679672851,58766.19470708726,26536.37582433423,74258.18862452936,51565.339062848194,36518.23927515308,63824.75167886689,49818.66578230918,31214.098333803213,55454.47823937452,43688.5730947842,10936.792150592984,56551.07827600264,56064.421324425464,67887.61268716963,23321.186239902247,54794.777388784656,55215.14725307505,42691.57153829672,60190.59924434755,59994.68751363818,56938.86939280927,24112.89906814105,56930.913398778444,41037.347079625375,76083.49495696368,64682.87076172693,51279.77213847536,37876.0141753861,37543.333901651524,57837.71157898636,56275.97331333026,71023.97889419644,44240.43092071472,79935.012346759477,58017.584727208254,45189.98561664212,76928.17033868484,56323.79915921683,40741.20220179615,41625.473690809544,61684.04755268171,56866.592992227829,75117.38650021306,60031.12483388769,44193.50446288398,60433.066363486665,45590.49578503727,44644.56198790086,57509.985820718146,60482.426303038024,57380.28271989199,42109.4773599863,37058.36353935319,29510.664331991986,68179.159780796,56129.792172210065,29709.945712346984,27164.612206731632,40561.04909460817,73005.91564874748,41962.978054215535,24389.630636105798,47661.52733094999,44603.470313163714,35642.73627657436,55156.81822324828,49270.214726443235,50491.9549311738,46543.9

8928691349,27552.061173648814,40249.64098389653,48748.43045314459,2825
5.321704522048,34940.63941877752,51040.16505554376,41799.79873119102,7
0106.91577685522,62445.98626818762,62170.948317373884,32776.0506687107
26,36237.457684786044,54072.42966238472,58272.144128592845,55860.43881
392216,27033.387461664166,57713.821340267525,58580.86162571216,58469.0
88888404,49646.71773688643,47384.68613145746,48027.911064924985,43448.
53644142674,70721.8111803573,60345.61707867791,53952.30822996589,54413
.3619352024,38243.51501620405,48992.32564925967,39263.60016333872,4129
9.201379449354,57080.03406378911,61140.23627539584,31435.063665090533,
38550.70843732128,60249.93489216604,66094.67071649017,85458.0874192904
2,38210.209510475266,29284.330952693377,46537.00439232847,71798.909949
81551,29920.91300415127,38500.15118324978,41756.6429613713,62893.81830
469703,21204.902657035156,49792.433808586444,42603.15766400342,62198.0
7780121966,45803.597472562484,55100.76941853335,58559.19495829297,6452
2.769907992915,33722.742101633026,37618.842050315674,30163.13064884644
7,58233.25752976046,51288.39509439996,16710.494408786974,46553.0055413
7209,37228.903486095296,52628.17128362768,94778.88503905115,55512.2249
8211457,57240.31830813819,80848.15347244313,43739.49779592374,45573.65
1286431406,64637.250790170634,31314.239353304234,45372.75817316271,864
00.74729518284,29335.73748868644,65320.74943144098,81326.64159304887,2
1801.134222914323,55844.21266056356,10463.783914746833,50050.632598957
42,63864.05209306745,34791.56071561194,53830.756335114755,59767.043627
354855,62876.76674984998,50573.56992440043,45504.85027989819,33320.494
61338755,64952.881601987116,54209.54606735204,55837.38793501893,46955.
290439065146,37999.421534035304,49357.649280326375,62726.46323307927,1
6296.659990230924,40899.49557852055,68001.18434640292,21851.7094093718
88,40469.56416761556,40634.81915430164,69123.12561918084,59772.8298700
9259,46977.626990787234,60025.107330771214,56859.20606922862,71156.887
17805257,56146.44193939246,37083.68134739644,43354.339683072765,54669.
63010164327,75499.3606186457,66073.14728352329,64119.65871151225,34512
.141447086906,46729.31156911439,37285.49107071373,40218.66667110409,33
655.50443986418,44442.37852114956,28916.49471232037,63534.15760679971]
,"yaxis":"y"}],"layout":{"legend":{"itemsizing":"constant","title":
{"text":"Genere"},"tracegroupgap":0},"margin":{"t":60},"template":
{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],

```
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
```

```
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
```

{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"linecolor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF6","caxis":{"gridcolor":"white","linecolor":"white","ticks":""}},"title":{"x":5.0e-2},"xaxis":{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"","title":{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"","title":{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":{"text":"Grafico a dispersione interattivo"},"xaxis":{"anchor":"y","domain":[0,1],"title":{"text":"Età"}},"yaxis":{"anchor":"x","domain":[0,1],"title":{"text":"Reddito"}}}}

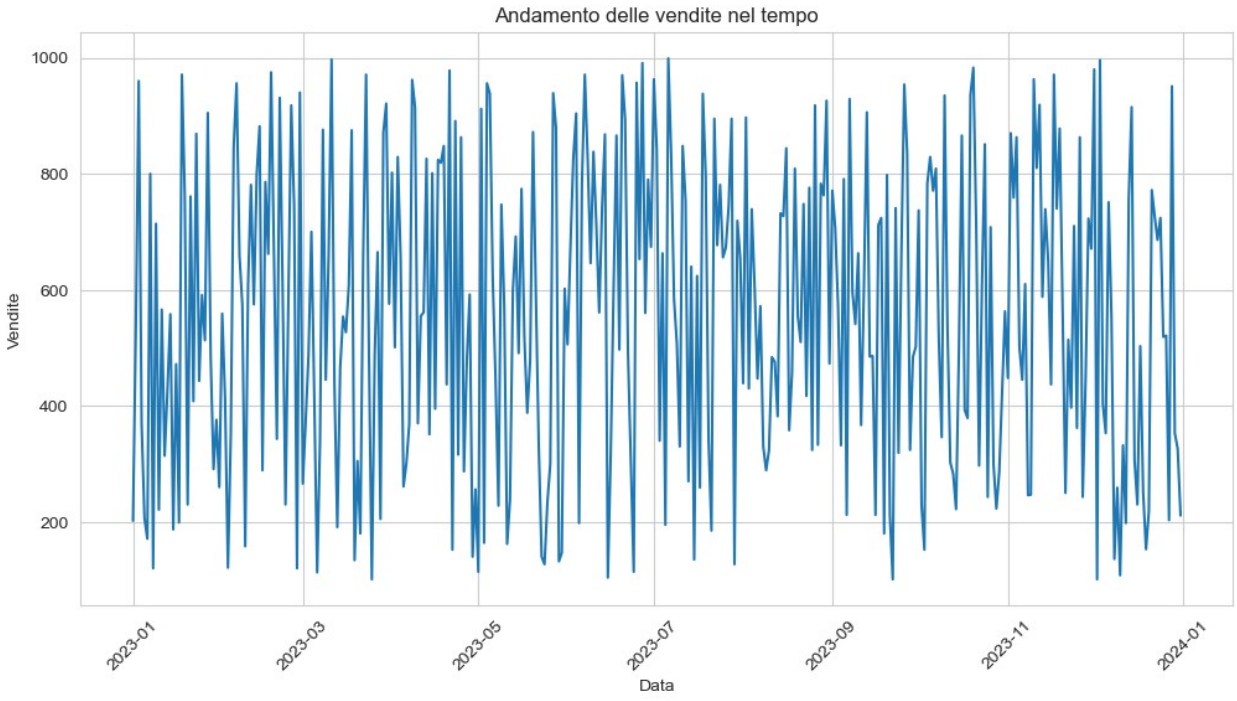
1. inizializziamo il dataframe

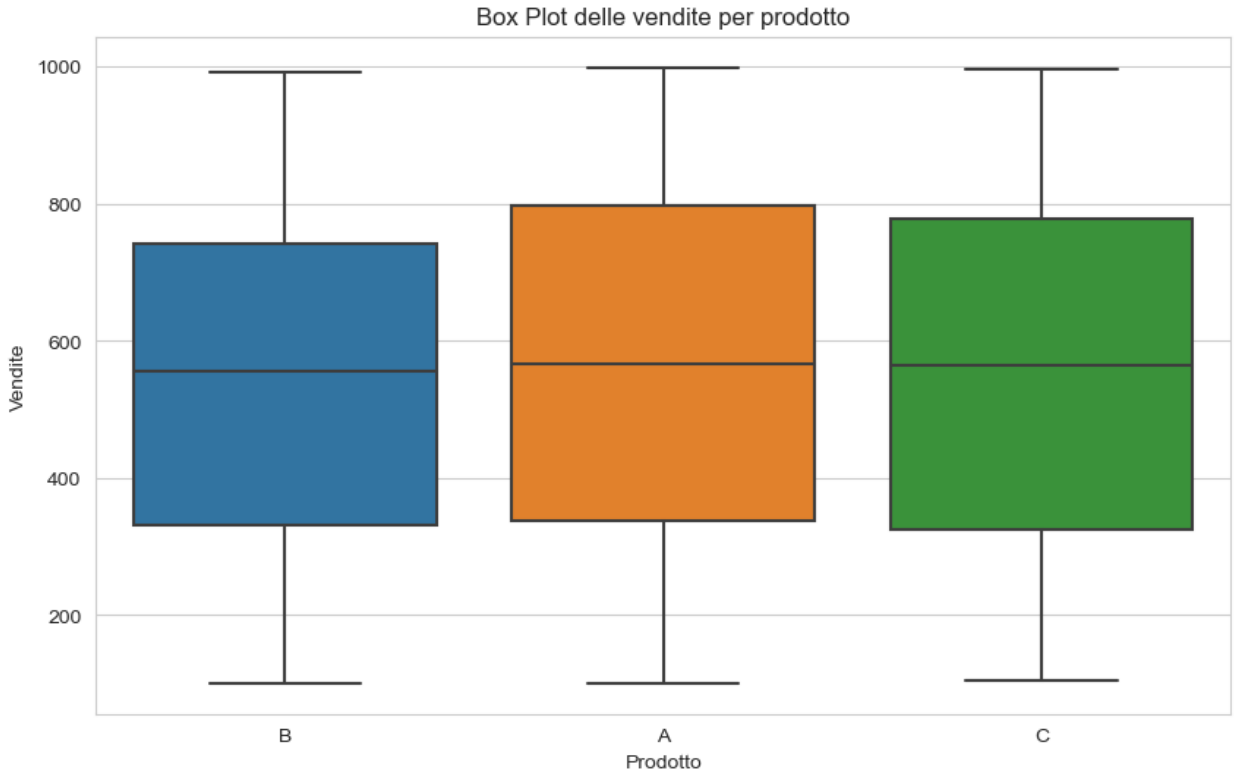
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Genera dati casuali per l'esplorazione
np.random.seed(42)
data = {
    'Data': pd.date_range(start='2023-01-01', end='2023-12-31', freq='D'), #data casuale
    'Vendite': np.random.randint(100, 1000, size=365), #100 numeri casuali tra1 100 e 1000
    'Prodotto': np.random.choice(['A', 'B', 'C'], size=365) #
}

df = pd.DataFrame(data)

# Visualizza le prime righe del dataset
print(df.head())
```

```
        Data  Vendite Prodotto
0 2023-01-01      202        B
1 2023-01-02      535        A
2 2023-01-03      960        C
3 2023-01-04      370        A
4 2023-01-05      206        A
```

1. facciamo un analisi tramite grafico a linee o a box (o quello che si più addice ai dati)

```
# Visualizza un grafico delle vendite nel tempo
plt.figure(figsize=(12, 6))
sns.lineplot(x='Data', y='Vendite', data=df)
plt.title('Andamento delle vendite nel tempo')
plt.xlabel('Data')
plt.ylabel('Vendite')
plt.xticks(rotation=45)
plt.show()

# Visualizza una box plot delle vendite per prodotto
plt.figure(figsize=(10, 6))
sns.boxplot(x='Prodotto', y='Vendite', data=df)
plt.title('Box Plot delle vendite per prodotto')
plt.xlabel('Prodotto')
plt.ylabel('Vendite')
plt.show()
```





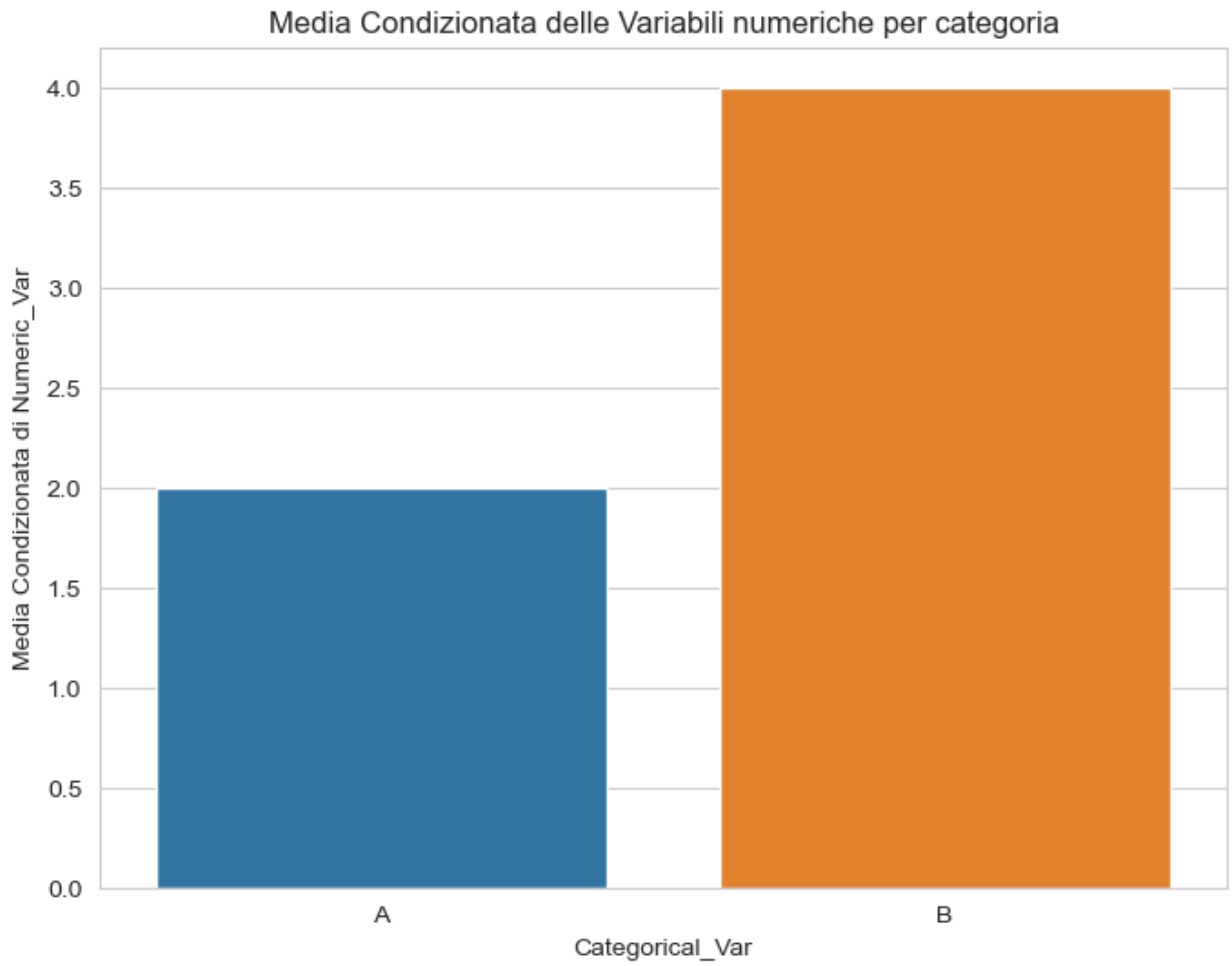

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Genera dati di esempio
data = {
    'Numeric_Var': [1, 2, 3, 4, np.nan, 6],
    'Categorical_Var': ['A', 'B', 'A', 'B', 'A', 'B']
}

# Crea un DataFrame
df = pd.DataFrame(data)
print(df)
```

```
   Numeric_Var Categorical_Var
0          1.0               A
1          2.0               B
2          3.0               A
3          4.0               B
4          NaN               A
5          6.0               B
```

```python
#calcola la media condizionata
conditional_means =
df['Numeric_Var'].fillna(df.groupby('Categorical_Var')
```

```
['Numeric_Var'].transform('mean'))

#aggiorna la colonna numeric_var con la media condizionata
df['Numeric_Var'] = conditional_means
print(df)

#crea un graico a barre per mostrare la sedia condizionata per ogni
categoria
plt.figure(figsize=(8,6))
sns.barplot(data=df, x='Categorical_Var', y='Numeric_Var',
errorbar=None) #errorbar = ci (confident interval)
plt.xlabel('Categorical_Var')
plt.ylabel('Media Condizionata di Numeric_Var')
plt.title('Media Condizionata delle Variabili numeriche per
categoria')
plt.show()
```

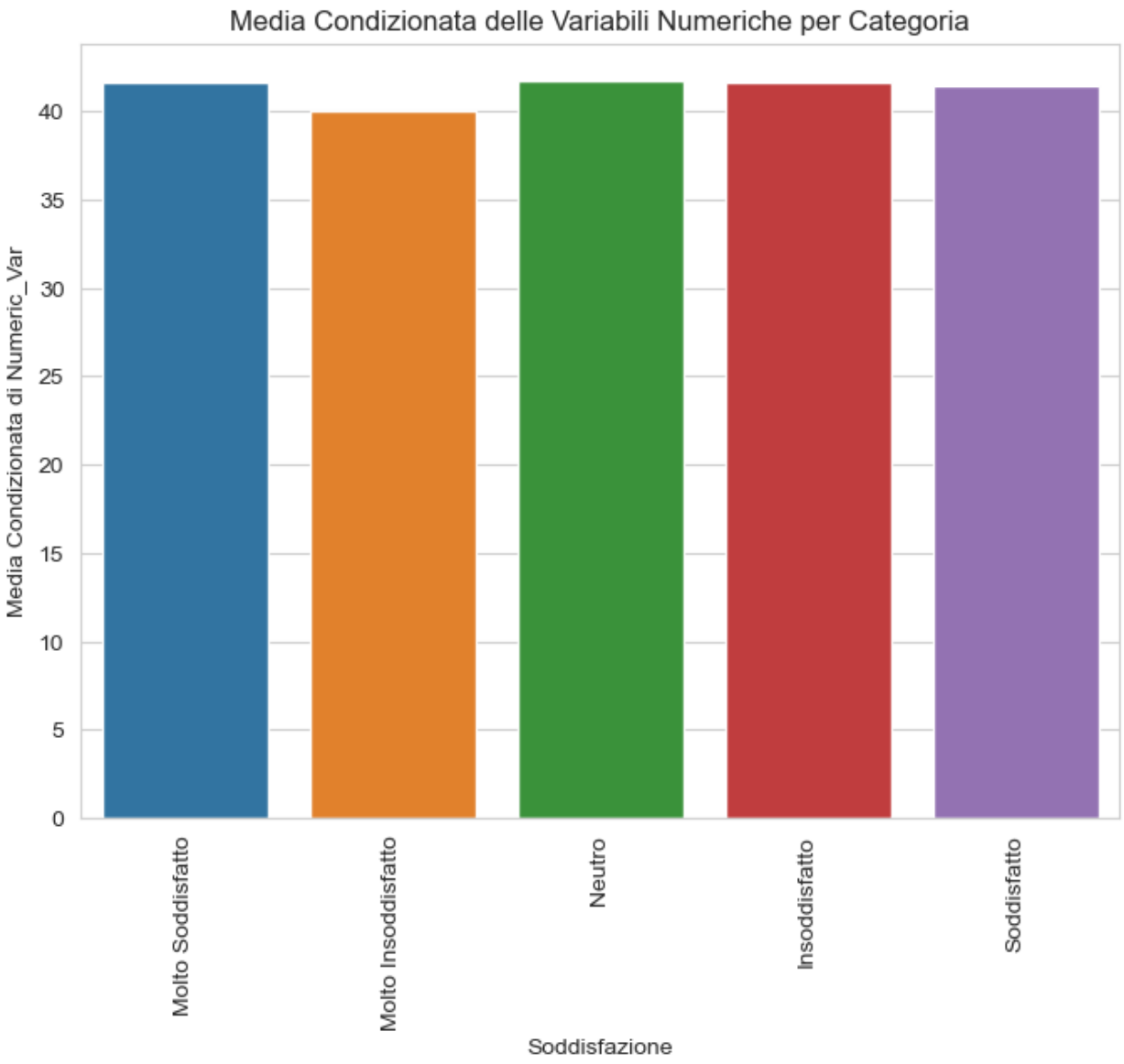
```
   Numeric_Var Categorical_Var
0          1.0               A
1          2.0               B
2          3.0               A
3          4.0               B
4          2.0               A
5          6.0               B
```




```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Genera dati casuali per l'esplorazione
np.random.seed(42)
data = {
    'Età': np.random.randint(18, 65, size=500),
    'Soddisfazione': np.random.choice(['Molto Soddisfatto',
'Soddisfatto', 'Neutro', 'Insoddisfatto', 'Molto Insoddisfatto'],
size=500)
}

df = pd.DataFrame(data)
print(df)
conditional_means = df.groupby('Soddisfazione')
['Età'].transform('mean')

df['Numeric_Var'] = conditional_means
```

```
print(df)

# Crea un grafico a barre per mostrare la media condizionata per ogni categoria
plt.figure(figsize=(8, 6))
sns.barplot(data=df, x='Soddisfazione', y='Numeric_Var', errorbar=None)
plt.xlabel('Soddisfazione')
plt.ylabel('Media Condizionata di Numeric_Var')
plt.title('Media Condizionata delle Variabili Numeriche per Categoria')
plt.xticks(rotation=90)

plt.show()
```

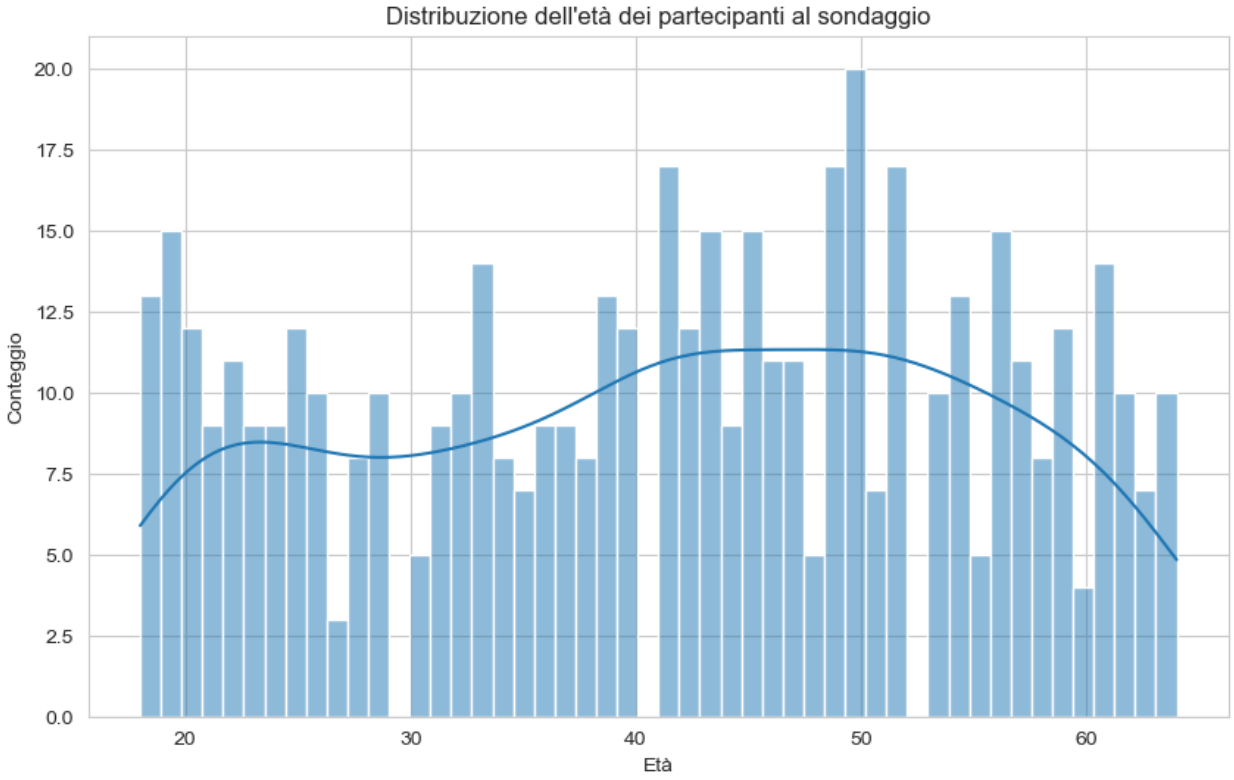
```
     Età        Soddisfazione
0     56    Molto Soddisfatto
1     46  Molto Insoddisfatto
2     32               Neutro
3     60               Neutro
4     25  Molto Insoddisfatto
..   ...                  ...
495   37    Molto Soddisfatto
496   41    Molto Soddisfatto
497   29    Molto Soddisfatto
498   52    Molto Soddisfatto
499   50    Molto Soddisfatto

[500 rows x 2 columns]
     Età        Soddisfazione  Numeric_Var
0     56    Molto Soddisfatto    41.651376
1     46  Molto Insoddisfatto    40.054054
2     32               Neutro    41.747368
3     60               Neutro    41.747368
4     25  Molto Insoddisfatto    40.054054
..   ...                  ...          ...
495   37    Molto Soddisfatto    41.651376
496   41    Molto Soddisfatto    41.651376
497   29    Molto Soddisfatto    41.651376
498   52    Molto Soddisfatto    41.651376
499   50    Molto Soddisfatto    41.651376

[500 rows x 3 columns]
```

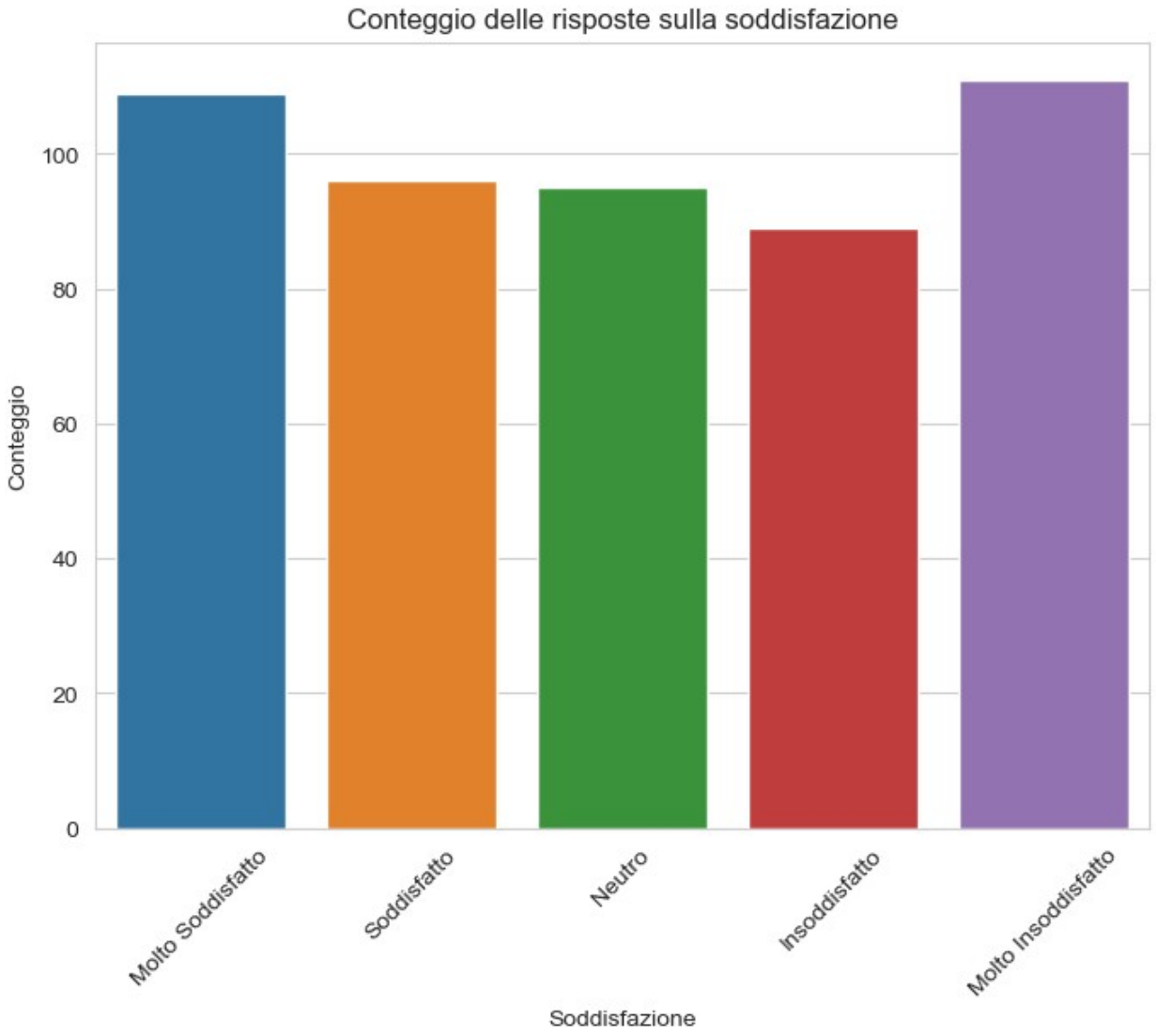



```
# Visualizza le prime righe del dataset
print(df.head())

# Visualizza una distribuzione dell'età
plt.figure(figsize=(10, 6))
sns.histplot(df['Età'], bins=50, kde=True)
plt.title('Distribuzione dell\'età dei partecipanti al sondaggio')
plt.xlabel('Età')
plt.ylabel('Conteggio')
plt.show()

# Visualizza un conteggio delle risposte sulla soddisfazione
plt.figure(figsize=(8, 6))
sns.countplot(x='Soddisfazione', data=df, order=['Molto Soddisfatto',
```

```
'Soddisfatto', 'Neutro', 'Insoddisfatto', 'Molto Insoddisfatto'])
plt.title('Conteggio delle risposte sulla soddisfazione')
plt.xlabel('Soddisfazione')
plt.ylabel('Conteggio')
plt.xticks(rotation=45)
plt.show()
```

	Età	Soddisfazione	Numeric_Var
0	56	Molto Soddisfatto	41.651376
1	46	Molto Insoddisfatto	40.054054
2	32	Neutro	41.747368
3	60	Neutro	41.747368
4	25	Molto Insoddisfatto	40.054054

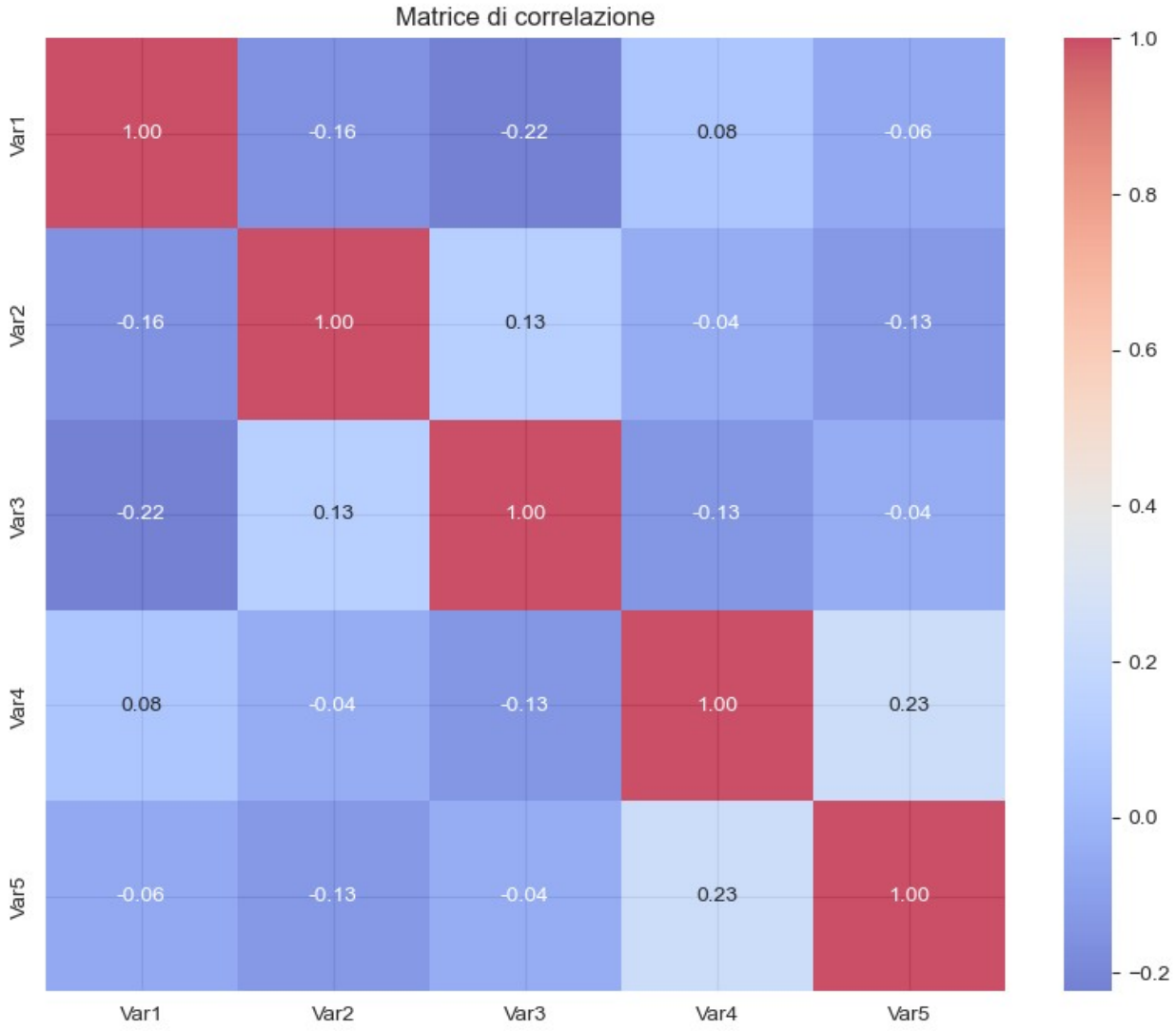





```
## import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Genera un dataset di esempio con variabili numeriche
np.random.seed(42)
data = pd.DataFrame(np.random.rand(100, 5), columns=['Var1', 'Var2',
'Var3', 'Var4', 'Var5'])

# Aggiungi alcune variabili categoriche generate casualmente
data['Categoria1'] = np.random.choice(['A', 'B', 'C'], size=100)
data['Categoria2'] = np.random.choice(['X', 'Y'], size=100)

#calcola la matrice di correlazione tra tutte le variabili numerici
(correlazione = )
correlation_matrix = data.corr()
```

```
#visualizza la matrice di correlazione come heatmap
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f", alpha=0.7)

plt.title('Matrice di correlazione')
plt.show()
```

```
/var/folders/hm/gkr5jwq528304nzg_2t2dgq00000gn/T/
ipykernel_1347/1612240019.py:14: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In
a future version, it will default to False. Select only valid columns
or specify the value of numeric_only to silence this warning.
```



```python
import pandas as pd
import numpy as np

# Impostare il seed per rendere i risultati riproducibili
np.random.seed(41)

# Creare un dataframe vuoto
df = pd.DataFrame()

# Generare dati casuali
n_rows = 10000
df['CatCol1'] = np.random.choice(['A', 'B', 'C'], size=n_rows)
df['CatCol2'] = np.random.choice(['X', 'Y'], size=n_rows)
df['NumCol1'] = np.random.randn(n_rows)
df['NumCol2'] = np.random.randint(1, 100, size=n_rows)
df['NumCol3'] = np.random.uniform(0, 1, size=n_rows)

# Calcolare il numero totale di missing values desiderati
total_missing_values = int(0.03 * n_rows * len(df.columns))

# Introdurre missing values casuali
for column in df.columns:
    num_missing_values = np.random.randint(0, total_missing_values +
1)
    missing_indices = np.random.choice(n_rows,
size=num_missing_values, replace=False)
    df.loc[missing_indices, column] = np.nan
df
```

```
     CatCol1 CatCol2   NumCol1  NumCol2   NumCol3
0          A     NaN  0.440877     49.0  0.246007
1          A       Y  1.945879     28.0  0.936825
2          C       X  0.988834     42.0  0.751516
3          A       Y -0.181978     73.0  0.950696
4          B       X  2.080615     74.0  0.903045
...      ...     ...       ...      ...       ...
9995       C       Y  1.352114     61.0  0.728445
9996       C       Y  1.143642     67.0  0.605930
9997       A       X -0.665794     54.0  0.071041
9998       C       Y  0.004278      NaN       NaN
9999       A       X  0.622473     95.0  0.751384

[10000 rows x 5 columns]
```

```python
#identificazione delle righe con dati mancati
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
len(righe_con_dati_mancanti)
```

```
3648
```

```
missing_percent = (df.isnull().sum() / len(df) * 100)
missing_percent
```

```
CatCol1     0.29
CatCol2    10.63
NumCol1     9.67
NumCol2    10.48
NumCol3    12.69
dtype: float64
```

```
missing_matrix = df.isnull()
#crea una heatmap colorata
plt.figure(figsize=(8,6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
plt.title('Matice di missing values')
plt.show()
```

I missing values

Immaginate di avere una lista di dati, come altezza di persone o temperatura di città. A volte, però, alcuni dati possono mancare o essere incompleti. Questi sono i "missing values"! Ad esempio, se stiamo misurando l'altezza di alcune persone e per una persona non abbiamo questa informazione, abbiamo un "missing value".

Metodo dell'eliminazione: Immagina di avere una lista di persone con informazioni su nome, età e altezza. Se una persona ha almeno una di queste informazioni mancante, invece di cercare di riempire solo quel dato mancante, puoi scegliere di buttare via l'intera riga. È come se, se anche solo una parte del biglietto di un concerto manca, non ti lasciano entrare affatto.

Questo metodo semplifica l'analisi perché non devi preoccuparti dei singoli dati mancanti. Ma, c'è il rischio che buttando via tutta la riga, possiamo perdere informazioni importanti su altre cose. Ad esempio, se buttiamo via l'intera riga solo perché non sappiamo l'età di qualcuno, potremmo perdere anche informazioni su nome e altezza che potrebbero essere utili.

Metodo del riempimento: Un'altra strategia è quella di riempire i "missing values" con dei valori approssimati o medi. Questo significa sostituire i dati mancanti con altri dati simili o con la media degli altri dati disponibili. Ad esempio, se non conosciamo l'altezza di una persona, potremmo decidere di usare l'altezza media delle altre persone nel nostro gruppo.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Genera dati di esempio
data = {
    'Numeric_Var': [1, 2, 3, 4, np.nan, 6],
    'Categorical_Var': ['A', 'B', 'A', 'B', 'A', 'B']
}

# Crea un DataFrame
df = pd.DataFrame(data)
print(df)
```

```
   Numeric_Var Categorical_Var
0          1.0              A
1          2.0              B
2          3.0              A
3          4.0              B
4          NaN              A
5          6.0              B
```

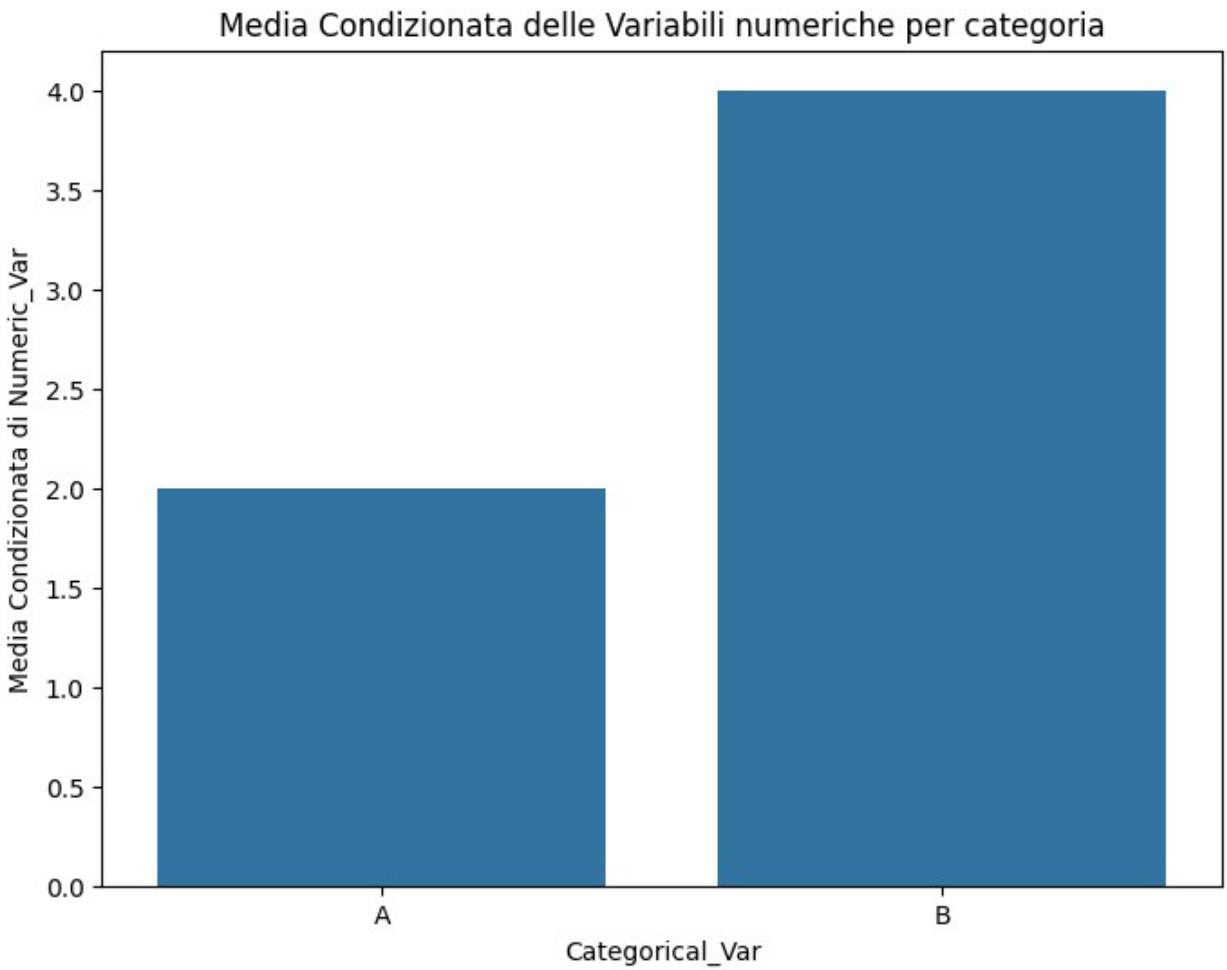
1. **Importazione delle librerie**:
   - `import pandas as pd`: Importa la libreria Pandas con l'alias `pd`.
   - `import matplotlib.pyplot as plt`: Importa il modulo `pyplot` dalla libreria Matplotlib con l'alias `plt`.

- `import numpy as np`: Importa la libreria NumPy con l'alias `np`.
- `import seaborn as sns`: Importa la libreria Seaborn con l'alias `sns`.

2. **Generazione dei dati di esempio**:
   - Viene creato un dizionario `data` che contiene due chiavi: 'Numeric_Var' e 'Categorical_Var'. La chiave 'Numeric_Var' contiene una lista di valori numerici, mentre la chiave 'Categorical_Var' contiene una lista di valori categorici.
3. **Creazione del DataFrame**:
   - Viene creato un DataFrame `df` utilizzando il dizionario `data`, dove le chiavi diventano i nomi delle colonne e i valori diventano i dati nel DataFrame.
4. **Stampa del DataFrame**:
   - Viene utilizzata la funzione `print(df)` per stampare il DataFrame `df`, mostrando così i dati generati.

```
#calcola la media condizionata
conditional_means =
df['Numeric_Var'].fillna(df.groupby('Categorical_Var')
['Numeric_Var'].transform('mean'))

#aggiorna la colonna numeric_var con la media condizionata
df['Numeric_Var'] = conditional_means
print(df)

#crea un graico a barre per mostrare la sedia condizionata per ogni
categoria
plt.figure(figsize=(8,6))
sns.barplot(data=df, x='Categorical_Var', y='Numeric_Var',
errorbar=None) #errorbar = ci (confident interval)
plt.xlabel('Categorical_Var')
plt.ylabel('Media Condizionata di Numeric_Var')
plt.title('Media Condizionata delle Variabili numeriche per
categoria')
plt.show()
```

```
   Numeric_Var Categorical_Var
0          1.0              A
1          2.0              B
2          3.0              A
3          4.0              B
4          2.0              A
5          6.0              B
```



1. **Calcolo della media condizionata**:
   - Viene utilizzata la funzione `fillna()` per sostituire i valori mancanti nella colonna `Numeric_Var` con la media condizionata di `Numeric_Var` per ciascuna categoria di `Categorical_Var`. Questo viene ottenuto tramite la funzione `groupby()` che raggruppa i dati per `Categorical_Var` e `transform('mean')` che calcola la media per ogni gruppo.
   - Il risultato è assegnato alla variabile `conditional_means`.
2. **Aggiornamento della colonna `Numeric_Var`**:
   - La colonna `Numeric_Var` nel DataFrame `df` viene aggiornata con i valori della media condizionata calcolata in precedenza.
3. **Stampa del DataFrame aggiornato**:
   - Viene utilizzata la funzione `print(df)` per mostrare il DataFrame `df` con la colonna `Numeric_Var` aggiornata.
4. **Creazione di un grafico a barre**:
   - Viene creato un grafico a barre utilizzando la libreria Seaborn (`sns.barplot()`) per visualizzare la media condizionata della variabile numerica `Numeric_Var` per ogni categoria di `Categorical_Var`.

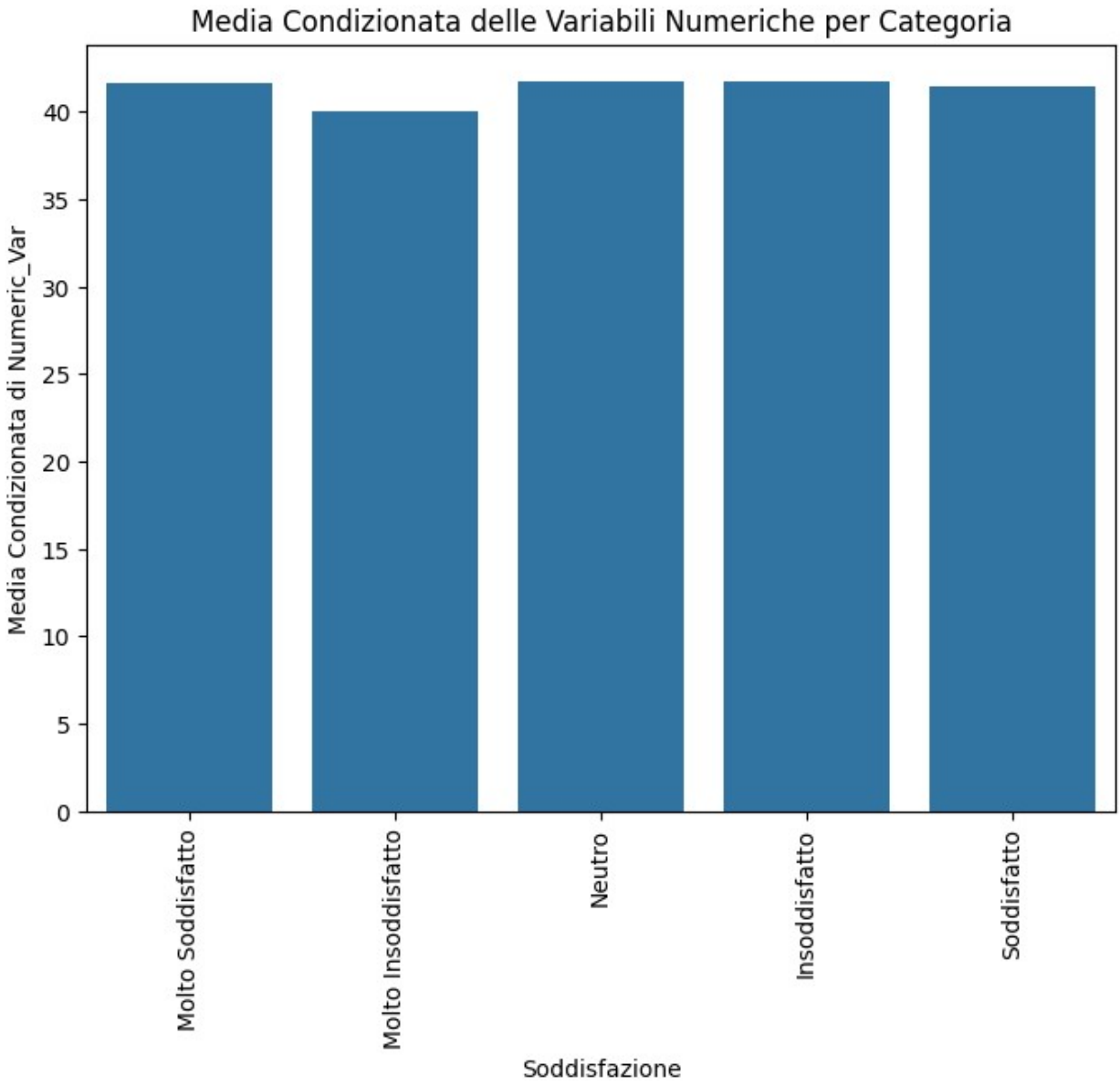
- Le barre rappresentano la media condizionata e vengono mostrate divise per categoria.
- Gli error bar non vengono visualizzati (`errorbar=None`).
- Viene aggiunta un'etichetta sull'asse x (`plt.xlabel('Categorical_Var')`) e sull'asse y (`plt.ylabel('Media Condizionata di Numeric_Var')`).
- Viene aggiunto un titolo al grafico (`plt.title('Media Condizionata delle Variabili numeriche per categoria')`).

5. **Visualizzazione del grafico**:
   - Il grafico a barre viene visualizzato utilizzando la funzione `plt.show()`.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Genera dati casuali per l'esplorazione
np.random.seed(42)
data = {
    'Età': np.random.randint(18, 65, size=500),
    'Soddisfazione': np.random.choice(['Molto Soddisfatto',
'Soddisfatto', 'Neutro', 'Insoddisfatto', 'Molto Insoddisfatto'],
size=500)
}

df = pd.DataFrame(data)
print(df)
conditional_means = df.groupby('Soddisfazione')
['Età'].transform('mean')

df['Numeric_Var'] = conditional_means
print(df)

# Crea un grafico a barre per mostrare la media condizionata per ogni
categoria
plt.figure(figsize=(8, 6))
sns.barplot(data=df, x='Soddisfazione', y='Numeric_Var',
errorbar=None)
plt.xlabel('Soddisfazione')
plt.ylabel('Media Condizionata di Numeric_Var')
plt.title('Media Condizionata delle Variabili Numeriche per
Categoria')
plt.xticks(rotation=90)

plt.show()
```

```
     Età        Soddisfazione
0     56    Molto Soddisfatto
1     46  Molto Insoddisfatto
2     32              Neutro
3     60              Neutro
```

```
4     25  Molto Insoddisfatto
..   ...                  ...
495   37    Molto Soddisfatto
496   41    Molto Soddisfatto
497   29    Molto Soddisfatto
498   52    Molto Soddisfatto
499   50    Molto Soddisfatto

[500 rows x 2 columns]
     Età        Soddisfazione  Numeric_Var
0     56    Molto Soddisfatto    41.651376
1     46  Molto Insoddisfatto    40.054054
2     32               Neutro    41.747368
3     60               Neutro    41.747368
4     25  Molto Insoddisfatto    40.054054
..   ...                  ...          ...
495   37    Molto Soddisfatto    41.651376
496   41    Molto Soddisfatto    41.651376
497   29    Molto Soddisfatto    41.651376
498   52    Molto Soddisfatto    41.651376
499   50    Molto Soddisfatto    41.651376

[500 rows x 3 columns]
```



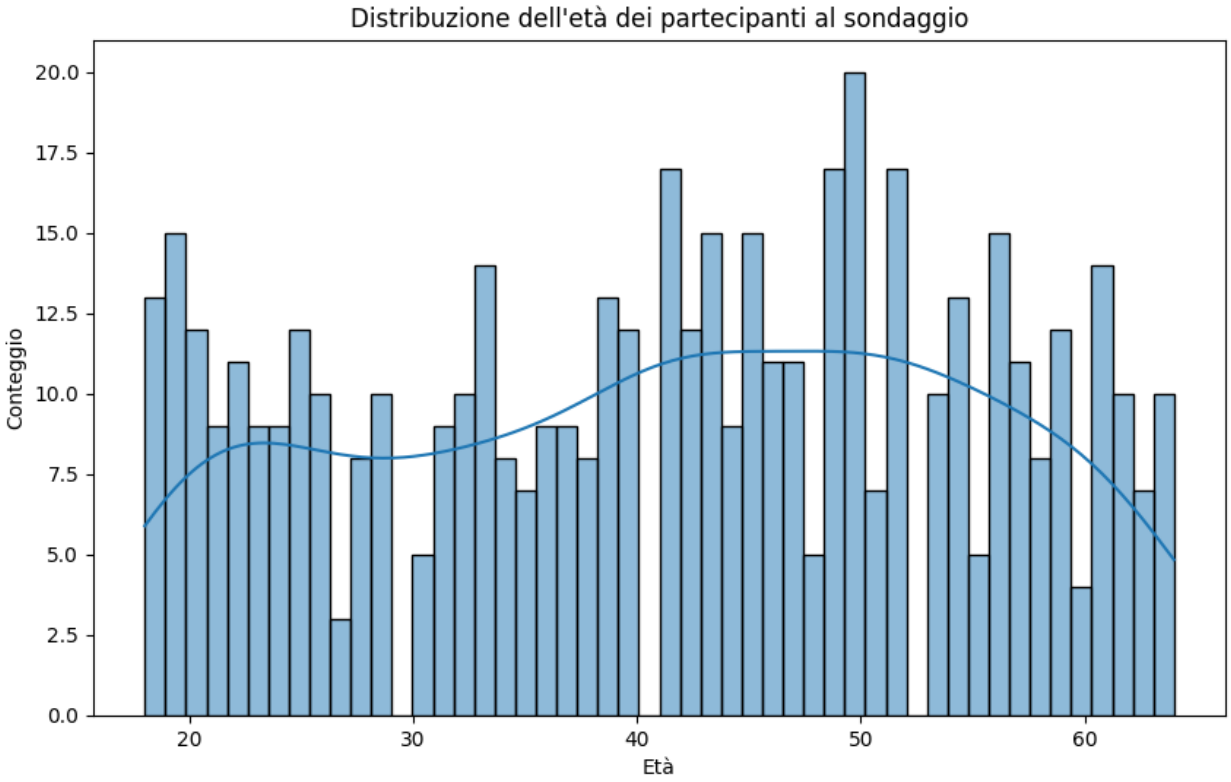
1. **Generazione dei dati casuali**:
   - Viene impostato il seed per la generazione di numeri casuali utilizzando `np.random.seed(42)`.
   - Viene creato un dizionario `data` che contiene due chiavi: 'Età' e 'Soddisfazione'. La chiave 'Età' contiene un array di età generate casualmente tra 18 e 65 anni utilizzando il metodo `np.random.randint()`. La chiave 'Soddisfazione' contiene un array di livelli di soddisfazione generati casualmente tra 'Molto Soddisfatto', 'Soddisfatto', 'Neutro', 'Insoddisfatto', e 'Molto Insoddisfatto' utilizzando il metodo `np.random.choice()`.
2. **Creazione del DataFrame**:
   - Viene creato un DataFrame `df` utilizzando il dizionario `data`, dove le chiavi diventano i nomi delle colonne e i valori diventano i dati nel DataFrame.
3. **Calcolo della media condizionata**:

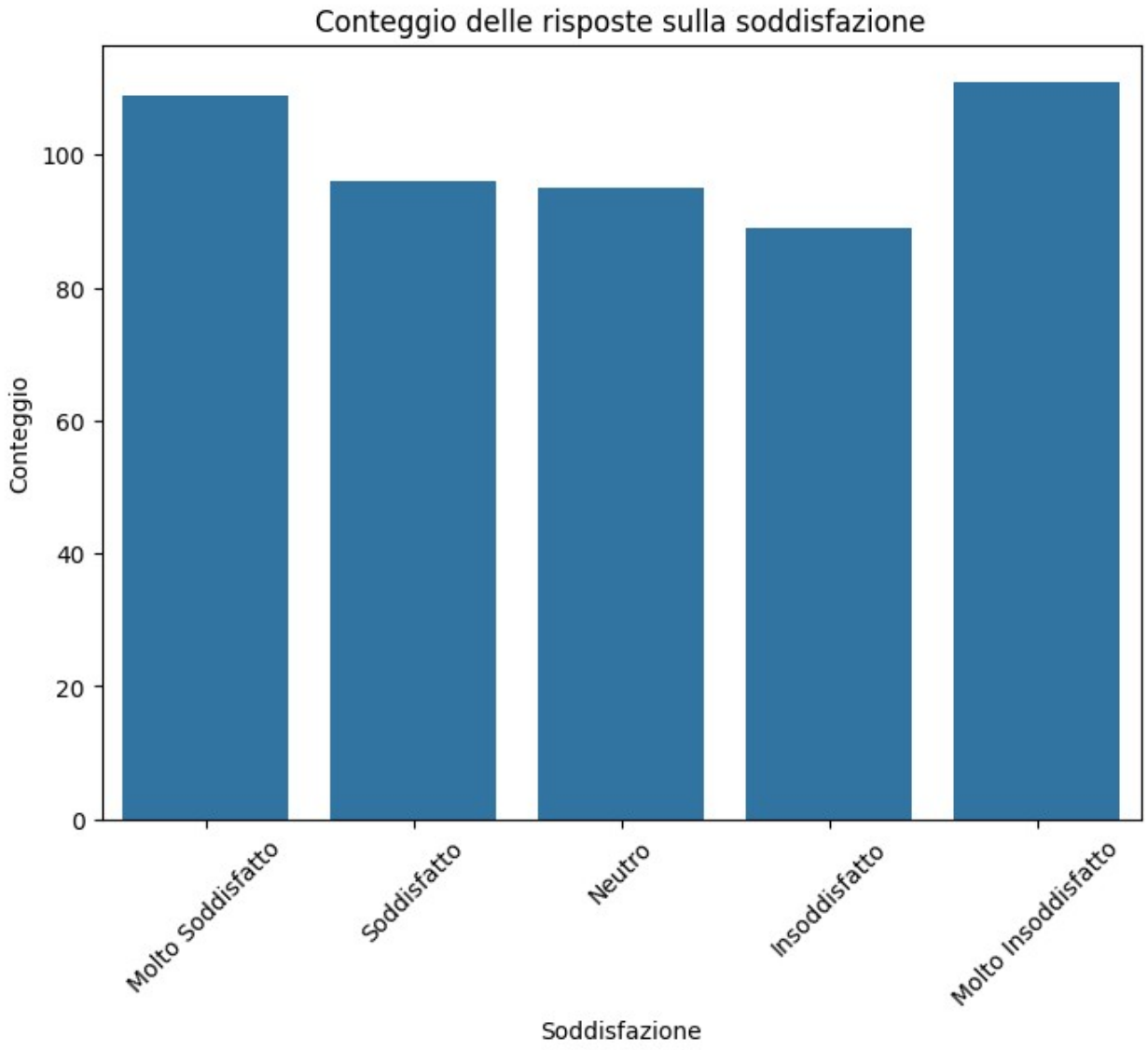
- Viene utilizzata la funzione `groupby()` per raggruppare i dati per livello di soddisfazione.
- Viene applicata la funzione `transform('mean')` per calcolare la media dell'età per ciascun gruppo di soddisfazione.
- I risultati sono assegnati alla variabile `conditional_means`.

4. **Aggiornamento del DataFrame**:
   - Viene creata una nuova colonna nel DataFrame `df` chiamata 'Numeric_Var' e riempita con i valori della media condizionata calcolata in precedenza.
5. **Creazione di un grafico a barre**:
   - Viene creato un grafico a barre utilizzando la libreria Seaborn (`sns.barplot()`) per visualizzare la media condizionata della variabile numerica 'Numeric_Var' per ogni categoria di 'Soddisfazione'.
   - Le barre rappresentano la media condizionata e vengono mostrate divise per categoria.
   - Gli error bar non vengono visualizzati (`errorbar=None`).
   - Viene aggiunta un'etichetta sull'asse x (`plt.xlabel('Soddisfazione')`) e sull'asse y (`plt.ylabel('Media Condizionata di Numeric_Var')`).
   - Viene aggiunto un titolo al grafico (`plt.title('Media Condizionata delle Variabili Numeriche per Categoria')`).
   - Le etichette sull'asse x vengono ruotate di 90 gradi per una migliore leggibilità (`plt.xticks(rotation=90)`).
6. **Visualizzazione del grafico**:
   - Il grafico a barre viene visualizzato utilizzando la funzione `plt.show()`.

```
# Visualizza le prime righe del dataset
print(df.head())

# Visualizza una distribuzione dell'età
plt.figure(figsize=(10, 6))
sns.histplot(df['Età'], bins=50, kde=True)
plt.title('Distribuzione dell\'età dei partecipanti al sondaggio')
plt.xlabel('Età')
plt.ylabel('Conteggio')
plt.show()

# Visualizza un conteggio delle risposte sulla soddisfazione
plt.figure(figsize=(8, 6))
sns.countplot(x='Soddisfazione', data=df, order=['Molto Soddisfatto',
'Soddisfatto', 'Neutro', 'Insoddisfatto', 'Molto Insoddisfatto'])
plt.title('Conteggio delle risposte sulla soddisfazione')
plt.xlabel('Soddisfazione')
plt.ylabel('Conteggio')
plt.xticks(rotation=45)
plt.show()

   Età        Soddisfazione  Numeric_Var
0   56    Molto Soddisfatto    41.651376
1   46  Molto Insoddisfatto    40.054054
```

```
2   32                Neutro    41.747368
3   60                Neutro    41.747368
4   25  Molto Insoddisfatto    40.054054
```





1. Visualizzazione delle prime righe del dataset

Il codice utilizza la funzione `print(df.head())` per mostrare le prime righe del dataset `df`, consentendo una rapida visualizzazione dei dati.

1. Distribuzione dell'età dei partecipanti al sondaggio

Viene creato un istogramma della distribuzione dell'età dei partecipanti al sondaggio utilizzando la libreria Seaborn. Di seguito sono i passaggi principali:

- Viene creato un nuovo grafico utilizzando `plt.figure(figsize=(10, 6))`, specificando le dimensioni della figura.
- Si utilizza `sns.histplot()` per creare l'istogramma della distribuzione dell'età (`df['Età']`). `bins=50` specifica il numero di bin utilizzati per l'istogramma, mentre `kde=True` aggiunge una stima della densità kernel.

- Si aggiungono titoli e etichette all'istogramma utilizzando le funzioni di `plt.title()`, `plt.xlabel()` e `plt.ylabel()`.
- Infine, il grafico viene mostrato utilizzando `plt.show()`.

1. Conteggio delle risposte sulla soddisfazione

Viene creato un grafico a barre per visualizzare il conteggio delle risposte sulla soddisfazione dei partecipanti al sondaggio. Di seguito sono i passaggi principali:

- Viene creato un nuovo grafico utilizzando `plt.figure(figsize=(8, 6))`, specificando le dimensioni della figura.
- Si utilizza `sns.countplot()` per creare il conteggio delle risposte sulla soddisfazione (`x='Soddisfazione'`). `order` specifica l'ordine delle categorie sull'asse x.
- Si aggiungono titoli e etichette al grafico utilizzando le funzioni di `plt.title()`, `plt.xlabel()` e `plt.ylabel()`.
- Le etichette sull'asse x vengono ruotate di 45 gradi per una migliore leggibilità utilizzando `plt.xticks(rotation=45)`.
- Infine, il grafico viene mostrato utilizzando `plt.show()`.

```
## import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Genera un dataset di esempio con variabili numeriche
np.random.seed(42)
data = pd.DataFrame(np.random.rand(100, 5), columns=['Var1', 'Var2',
'Var3', 'Var4', 'Var5'])

# Aggiungi alcune variabili categoriche generate casualmente
data['Categoria1'] = np.random.choice(['A', 'B', 'C'], size=100)
data['Categoria2'] = np.random.choice(['X', 'Y'], size=100)

#calcola la matrice di correlazione tra tutte le variabili numerici
(correlazione = )
correlation_matrix = data.corr()

#visualizza la matrice di correlazione come heatmap
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f", alpha=0.7)

plt.title('Matrice di correlazione')
plt.show()
```

```
---------------------------------------------------------------------
-----

ValueError                                Traceback (most recent call
last)

Cell In[9], line 14
```

```
     11 data['Categoria2'] = np.random.choice(['X', 'Y'], size=100)

     13 #calcola la matrice di correlazione tra tutte le variabili
numerici (correlazione = )

---> 14 correlation_matrix = data.corr()

     16 #visualizza la matrice di correlazione come heatmap

     17 plt.figure(figsize=(10,8))



File c:\Users\Utente\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\frame.py:10707, in DataFrame.corr(self, method,
min_periods, numeric_only)

  10705 cols = data.columns

  10706 idx = cols.copy()

> 10707 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)

  10709 if method == "pearson":

  10710     correl = libalgos.nancorr(mat, minp=min_periods)



File c:\Users\Utente\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\frame.py:1892, in DataFrame.to_numpy(self, dtype,
copy, na_value)

   1890 if dtype is not None:

   1891     dtype = np.dtype(dtype)

-> 1892 result = self._mgr.as_array(dtype=dtype, copy=copy,
na_value=na_value)

   1893 if result.dtype is not dtype:

   1894     result = np.array(result, dtype=dtype, copy=False)



File c:\Users\Utente\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\internals\managers.py:1656, in
BlockManager.as_array(self, dtype, copy, na_value)

   1654         arr.flags.writeable = False
```

```
   1655 else:

-> 1656     arr = self._interleave(dtype=dtype, na_value=na_value)

   1657     # The underlying data was copied within _interleave, so no
need

   1658     # to further copy if copy=True or setting na_value

   1660 if na_value is lib.no_default:


File c:\Users\Utente\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\internals\managers.py:1715, in
BlockManager._interleave(self, dtype, na_value)

   1713     else:

   1714         arr = blk.get_values(dtype)

-> 1715     result[rl.indexer] = arr

   1716     itemmask[rl.indexer] = 1

   1718 if not itemmask.all():


ValueError: could not convert string to float: 'B'
```

1. Generazione del Dataset e Calcolo della Matrice di Correlazione

Il codice fornito genera un dataset di esempio con variabili numeriche e categoriche, quindi calcola la matrice di correlazione tra le variabili numeriche e visualizza questa matrice come una heatmap utilizzando Seaborn.

1. Generazione del Dataset

- Viene importata la libreria NumPy con l'alias `np`, Seaborn con l'alias `sns`, e Matplotlib con l'alias `plt`.
- Viene generato un dataset di esempio `data` utilizzando `np.random.rand()` per generare valori casuali per 100 righe e 5 colonne, che vengono poi assegnate alle colonne 'Var1', 'Var2', 'Var3', 'Var4', e 'Var5'.
- Vengono aggiunte due variabili categoriche casuali al dataset: 'Categoria1' che contiene valori 'A', 'B', o 'C', e 'Categoria2' che contiene valori 'X' o 'Y'.

1. Calcolo della Matrice di Correlazione

- Viene calcolata la matrice di correlazione tra tutte le variabili numeriche utilizzando il metodo `.corr()` su `data`. Il risultato è assegnato alla variabile `correlation_matrix`.

1. Visualizzazione della Matrice di Correlazione

- Viene creato un nuovo grafico utilizzando `plt.figure(figsize=(10,8))`, specificando le dimensioni della figura.
- Si utilizza `sns.heatmap()` per visualizzare la matrice di correlazione come una heatmap. Gli argomenti `annot=True` mostrano i valori della correlazione all'interno delle celle, `cmap='coolwarm'` imposta la mappa dei colori a sfumature di blu e rosso, `fmt=".2f"` formatta i valori annotati con due cifre decimali, e `alpha=0.7` imposta la trasparenza della heatmap.
- Si aggiunge un titolo al grafico utilizzando `plt.title()`.
- Infine, il grafico viene mostrato utilizzando `plt.show()`.

```
import pandas as pd
import numpy as np

# Impostare il seed per rendere i risultati riproducibili
np.random.seed(41)

# Creare un dataframe vuoto
df = pd.DataFrame()

# Generare dati casuali
n_rows = 10000
df['CatCol1'] = np.random.choice(['A', 'B', 'C'], size=n_rows)
df['CatCol2'] = np.random.choice(['X', 'Y'], size=n_rows)
df['NumCol1'] = np.random.randn(n_rows)
df['NumCol2'] = np.random.randint(1, 100, size=n_rows)
df['NumCol3'] = np.random.uniform(0, 1, size=n_rows)

# Calcolare il numero totale di missing values desiderati
total_missing_values = int(0.03 * n_rows * len(df.columns))

# Introdurre missing values casuali
for column in df.columns:
    num_missing_values = np.random.randint(0, total_missing_values +
1)
    missing_indices = np.random.choice(n_rows,
size=num_missing_values, replace=False)
    df.loc[missing_indices, column] = np.nan
df
```

1. **Impostazione del seed**: Il seed di NumPy viene impostato a 41 per rendere i risultati riproducibili.

2. **Creazione del DataFrame**: Viene creato un DataFrame vuoto chiamato `df`.

3. **Generazione di dati casuali**: Viene generato un numero specificato di righe di dati casuali per le colonne `CatCol1`, `CatCol2`, `NumCol1`, `NumCol2`, e `NumCol3`. Le colonne `CatCol1` e `CatCol2` contengono valori casuali selezionati da un insieme di categorie, mentre le colonne `NumCol1`, `NumCol2`, e `NumCol3` contengono numeri casuali generati da diverse distribuzioni.

4. **Introduzione di valori mancanti**: Viene calcolato il numero totale di valori mancanti desiderati, corrispondente al 3% del totale dei valori nel DataFrame. Successivamente, vengono introdotti valori mancanti casuali nelle colonne del DataFrame. Per ciascuna colonna, viene scelto un numero casuale di valori mancanti, e vengono selezionati casualmente degli indici di riga dove impostare i valori mancanti utilizzando `np.nan`.

percentuale dei missing values

```
#identificazione delle righe con dati mancati
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
len(righe_con_dati_mancanti)
```

```
0
```

1. `df.isnull().any(axis=1)`: Questa parte del codice restituisce una serie booleana che indica per ogni riga se contiene almeno un valore mancante (`True`) o meno (`False`). Il metodo `isnull()` restituisce un DataFrame booleano con lo stesso indice e colonne del DataFrame originale, dove ogni valore è `True` se il valore corrispondente nel DataFrame originale è mancante (`NaN`), altrimenti `False`. Il metodo `any(axis=1)` restituisce `True` se almeno un valore nella riga è `True` (quindi se almeno un valore nella riga è mancante), altrimenti `False`.

2. `df[df.isnull().any(axis=1)]`: Questa parte del codice seleziona le righe del DataFrame `df` in cui almeno un valore è mancante. Utilizza la serie booleana restituita dalla parte precedente del codice per selezionare le righe corrispondenti.

3. `len(righe_con_dati_mancanti)`: Questa parte del codice calcola la lunghezza (il numero di righe) del DataFrame `righe_con_dati_mancanti`, cioè il numero di righe nel DataFrame `df` che contengono almeno un valore mancante. Il risultato rappresenta il conteggio delle righe con dati mancanti nel DataFrame originale.

```
missing_percent = (df.isnull().sum() / len(df) * 100)
missing_percent
```

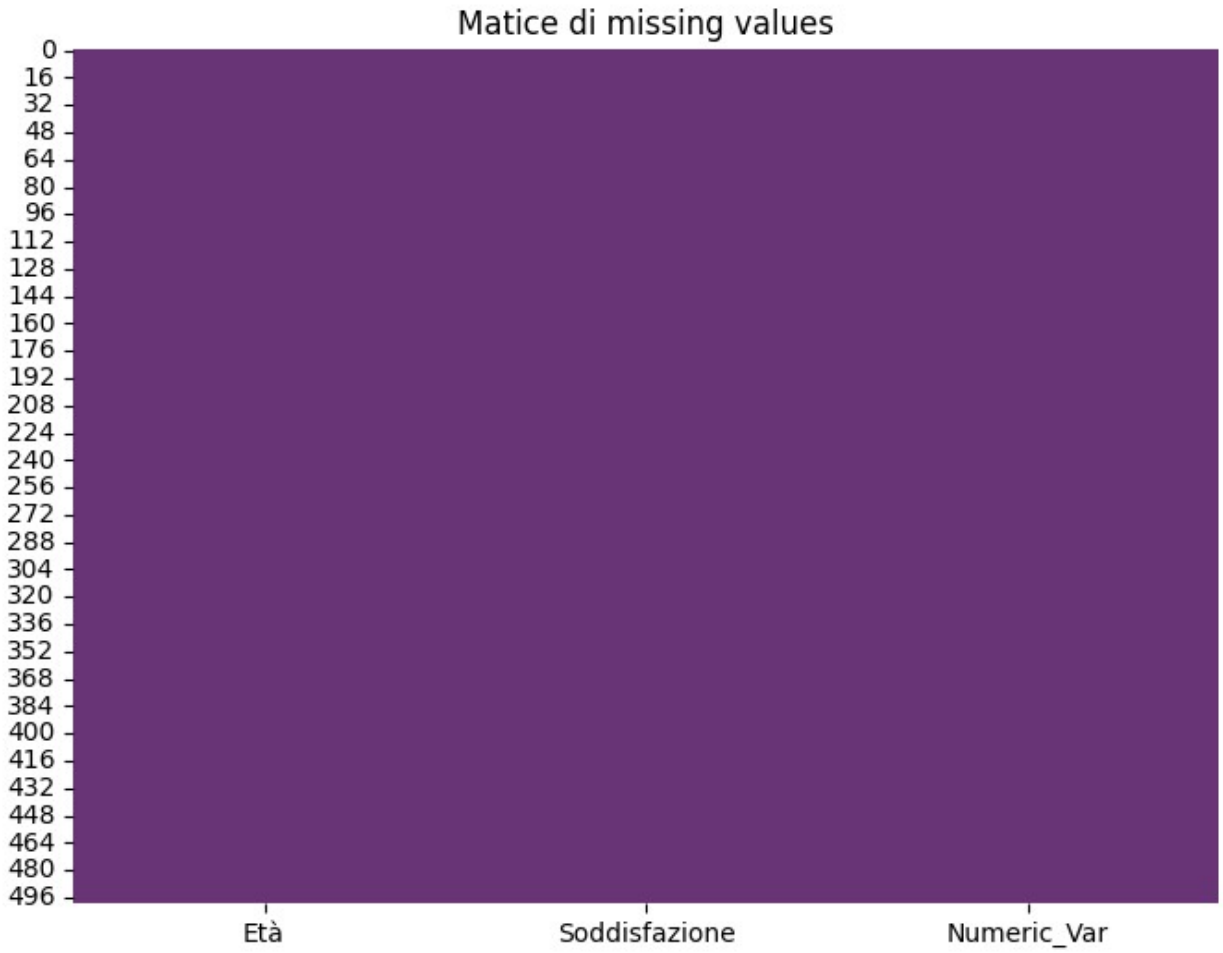
```
Età              0.0
Soddisfazione    0.0
Numeric_Var      0.0
dtype: float64
```

1. `df.isnull().sum()`: Questa parte del codice restituisce una Serie che contiene il numero di valori mancanti per ciascuna colonna del DataFrame `df`. Il metodo `isnull()` restituisce un DataFrame booleano con gli stessi indici e colonne del DataFrame originale, dove ogni valore è `True` se il valore corrispondente nel DataFrame originale è mancante (`NaN`), altrimenti `False`. Il metodo `sum()` calcola la somma dei valori `True` lungo l'asse 0 (per le colonne), restituendo il numero di valori mancanti per ciascuna colonna.

2. `len(df)`: Restituisce il numero totale di righe nel DataFrame `df`.

3. `df.isnull().sum() / len(df) * 100`: Questa parte del codice calcola la percentuale di valori mancanti per ciascuna colonna dividendo il numero di valori mancanti per ciascuna colonna (`df.isnull().sum()`) per il numero totale di righe nel DataFrame (`len(df)`), moltiplicando quindi per 100 per ottenere la percentuale.

```
missing_matrix = df.isnull()
#crea una heatmap colorata
plt.figure(figsize=(8,6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
plt.title('Matice di missing values')
plt.show()
```



1. `missing_matrix = df.isnull()`: Questa linea di codice crea una nuova variabile chiamata `missing_matrix` che contiene un DataFrame booleano con gli stessi indici e colonne del DataFrame originale `df`. Ogni valore nella `missing_matrix` è `True` se il valore corrispondente nel DataFrame originale è mancante (`NaN`), altrimenti è `False`.

2. `plt.figure(figsize=(8,6))`: Questo codice crea una nuova figura per il grafico con una dimensione di larghezza 8 pollici e altezza 6 pollici utilizzando `plt.figure(figsize=(8,6))`.

3. `sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)`: Questa parte del codice crea la heatmap utilizzando Seaborn (`sns`). Il parametro `missing_matrix` è la matrice dei valori mancanti ottenuta utilizzando `df.isnull()`. `cmap='viridis'` specifica la mappa dei colori da utilizzare per la heatmap. `cbar=False` rimuove la barra dei colori dalla heatmap. `alpha=0.8` imposta la trasparenza della heatmap.

4. `plt.title('Matrice di missing values')`: Aggiunge un titolo alla heatmap.

5. `plt.show()`: Mostra la heatmap.