Missing Values

1.0 Dataset

In [1]:
```
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)
df
```

Out[1]:

	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

1.1

In [2]:
```
df["punteggio"]
```

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

1.2 Righe con Dati Mancanti

In [3]:
```
# Identificazione delle righe con dati mancanti
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

Out[3]:

	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

1.3 Totale Dati Mancanti

In [4]:
```
#Conta quante righe con dati mancanti ci sono in totale
totale_dati_mancanti = righe_con_dati_mancanti.shape[0]
totale_dati_mancanti
```

Out[4]:
```
5
```

1.4

In [5]:
```
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
```

1.5 Creazione e Gestione di un DataFrame con Dati Mancanti in Pandas

In [6]:
```
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
```

Out[6]:

	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

1.6 Rimuovere Righe con Dati Mancanti

In [7]:
```
# Rimuovi le righe con dati mancanti
df1=df.dropna(inplace=False)
df1
```

Out[7]:

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

1.7 Generazione di un DataFrame con Dati di Esempio

In [8]:
```
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
```

Out[8]:

	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

1.8 “Gestione dei Valori Mancanti nelle Variabili Numeriche”

In [9]:
```
# Trattamento dei missing values nelle variabili numeriche
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols.columns
```

Out[9]:
```
Index(['Variable1', 'Variable2'], dtype='object')
```

1.9 Riempimento dei Valori Mancanti nelle Colonne Numeriche

In [10]:
```
df1[numeric_cols.columns] = df[numeric_cols.columns].fillna(df[numeric_cols.columns].mea
df1
```

Out[10]:

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

2.0 Gestione dei Valori Mancanti nelle Variabili Categoriche

In [11]:
```
# Trattamento dei missing values nelle variabili categoriche
categorial_cols = df.select_dtypes(exclude=['number'])
categorial_cols.columns
```

Out[11]:
```
Index(['Missing_Column'], dtype='object')
```

2.1 Riempi i Valori Mancanti nelle Colonne Categoriali con la Moda

In [12]:
```
df1[categorial_cols.columns] = df[categorial_cols.columns].fillna(df[categorial_cols.col
df1
```

Out[12]:

	Variable1	Variable2	Missing_Column
0	1	1.000000	A
1	2	2.000000	B
2	3	2.333333	A
3	4	4.000000	C
4	5	2.333333	A

2.2 DataFrame con e senza Valori Mancanti Sostituiti

In [13]:
```
print(f"il primo con i valori mancanti \n{df} \ne il secondo con i missing values sostit
```

```
il primo con i valori mancanti
   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 il secondo con i missing values sostituiti
   Variable1  Variable2 Missing_Column
0          1   1.000000              A
1          2   2.000000              B
2          3   2.333333              A
3          4   4.000000              C
4          5   2.333333              A
```

2.3 Analisi di Dati con Valori Mancanti

In [14]:
```
# Importa il modulo pandas per l'analisi dei dati
import pandas as pd
# Importa il modulo pyplot di matplotlib per la visualizzazione
import matplotlib.pyplot as plt
# Importa il modulo numpy per operazioni matematiche avanzate
import numpy as np

# Genera dati di esempio
data = {
# Lista dei valori per la feature 1
    'Feature1': [1, 2, np.nan, 4, 5],
# Lista dei valori per la feature 2
    'Feature2': [np.nan, 2, 3, 4, np.nan],
 # Lista dei valori per la feature 3
    'Feature3': [1, np.nan, 3, 4, 5]
}
# Crea un DataFrame
df = pd.DataFrame(data)
# Stampa il DataFrame
df
```

Out[14]:

	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

2.4 Conteggio dei Valori Mancanti per Colonna nel DataFrame

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
Feature1    1
Feature2    2
Feature3    1
dtype: int64
```

2.5

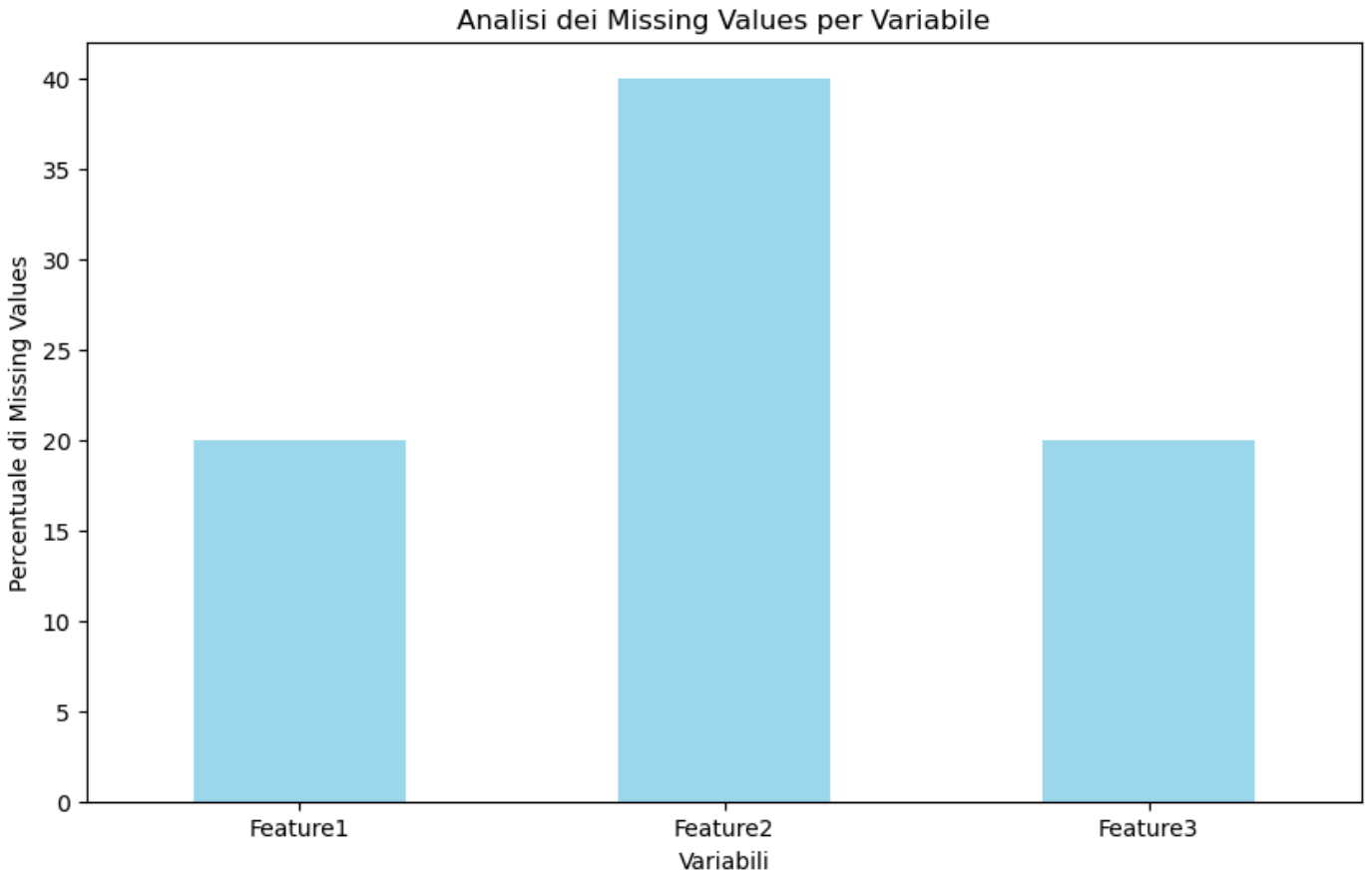
In [16]:
```
missing_percent = (df.isnull().sum() / len(df)) *100
missing_percent
```

Out[16]:
```
Feature1    20.0
Feature2    40.0
Feature3    20.0
dtype: float64
```

2.6 Analisi dei Missing Values per Variabile

In [17]:
```
# Calcola la percentuale di valori mancanti per ciascuna variabile
missing_percent = (df.isnull().sum() / len(df)) *100
```

```
# Crea il grafico a barre
# Imposta le dimensioni della figura
plt.figure(figsize=(10, 6))
# Crea il grafico a barre
missing_percent.plot(kind='bar', color='skyblue',alpha=0.8)
# Etichetta dell'asse x
plt.xlabel('Variabili')
# Etichetta dell'asse y
plt.ylabel('Percentuale di Missing Values')
# Titolo del grafico
plt.title('Analisi dei Missing Values per Variabile')
 # Ruota le etichette sull'asse x
plt.xticks(rotation=0)
# Mostra il grafico
plt.show()
```



2.7 Analisi della Presenza di Valori Mancanti nella Matrice dei Dati e Creazione di una Heatmap

In [18]:

```
# Import delle librerie necessarie
# Per la manipolazione dei dati tramite DataFrame
import pandas as pd
# Per creare visualizzazioni come la heatmap
import seaborn as sns
# Per personalizzare e mostrare i grafici
import matplotlib.pyplot as plt
# Per lavorare con dati numerici e mancanti

import numpy as np

# Genera dati di esempio
data = {
    'Feature1': [1, 2, np.nan, 4, 5],
    '2': [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
# True dove ci sono valori mancanti, False altrimenti
missing_matrix = df.isnull()
# Visualizza la matrice di missing values
missing_matrix
```

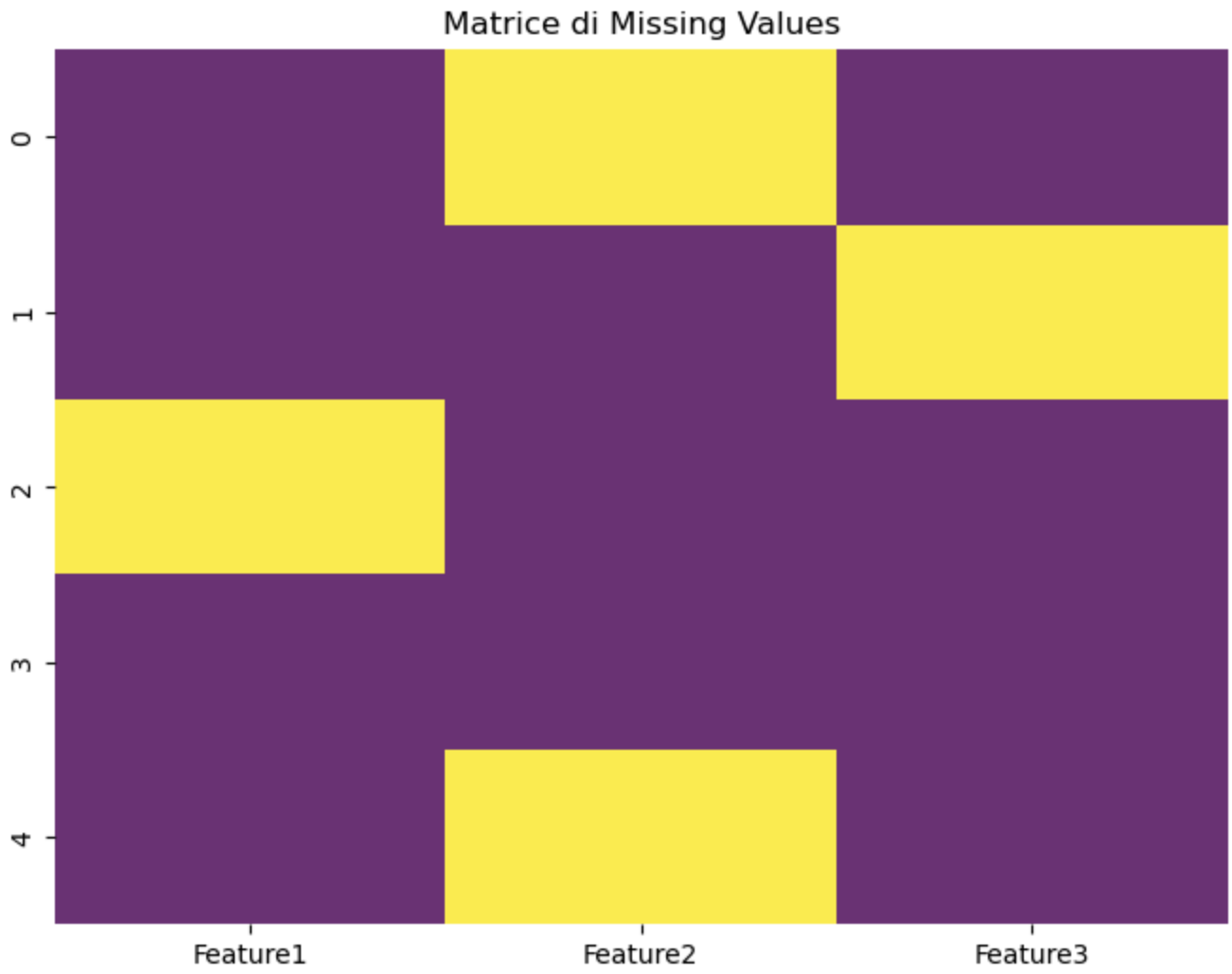
Out[18]:

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

2.8 Matrice di Missing Values

In [19]:
```
# Crea una 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
```

Out[19]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



2.9 Esplorazione Iniziale del DataSet con Dati Casuali

In [1]:

```
# Import delle librerie necessarieimport pandas as pd
# Per la manipolazione dei dati tramite DataFrame
import pandas as pd
# Per la generazione di dati casuali
import numpy as np
# Per la creazione di grafici
import matplotlib.pyplot as plt
# Per la creazione di visualizzazioni avanzate
import seaborn as sns
# Per la creazione di grafici interattivi
import plotly.express as px

# Genera dati casuali per l'esplorazione
np.random.seed(42)
data = {
     # Genera un array di età comprese tra 18 e 70 anni
    'Età': np.random.randint(18, 70, size=1000),
    # Genera un array di generi casuali
    'Genere': np.random.choice(['Maschio', 'Femmina'], size=1000),
    # Genera un array di punteggi casuali
    'Punteggio': np.random.uniform(0, 100, size=1000),
    # Genera un array di redditi casuali

    'Reddito': np.random.normal(50000, 15000, size=1000)
}
# Crea un DataFrame utilizzando i dati generati
df = pd.DataFrame(data)
```

```
# 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
```

3.0 Informazioni sul DataFrame e Statistiche Descrittive

In [21]:
```
print(df.info())
# Statistiche descrittive
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   int32
 1   Genere     1000 non-null   object
 2   Punteggio  1000 non-null   float64
 3   Reddito    1000 non-null   float64
dtypes: float64(2), int32(1), object(1)
memory usage: 27.5+ 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
```

3.1 Analisi dei Valori Mancanti nel DataFrame

In [22]:
```
# Gestione dei valori mancanti
missing_data = df.isnull().sum()
print("Valori mancanti per ciascuna colonna")
print(missing_data)
```

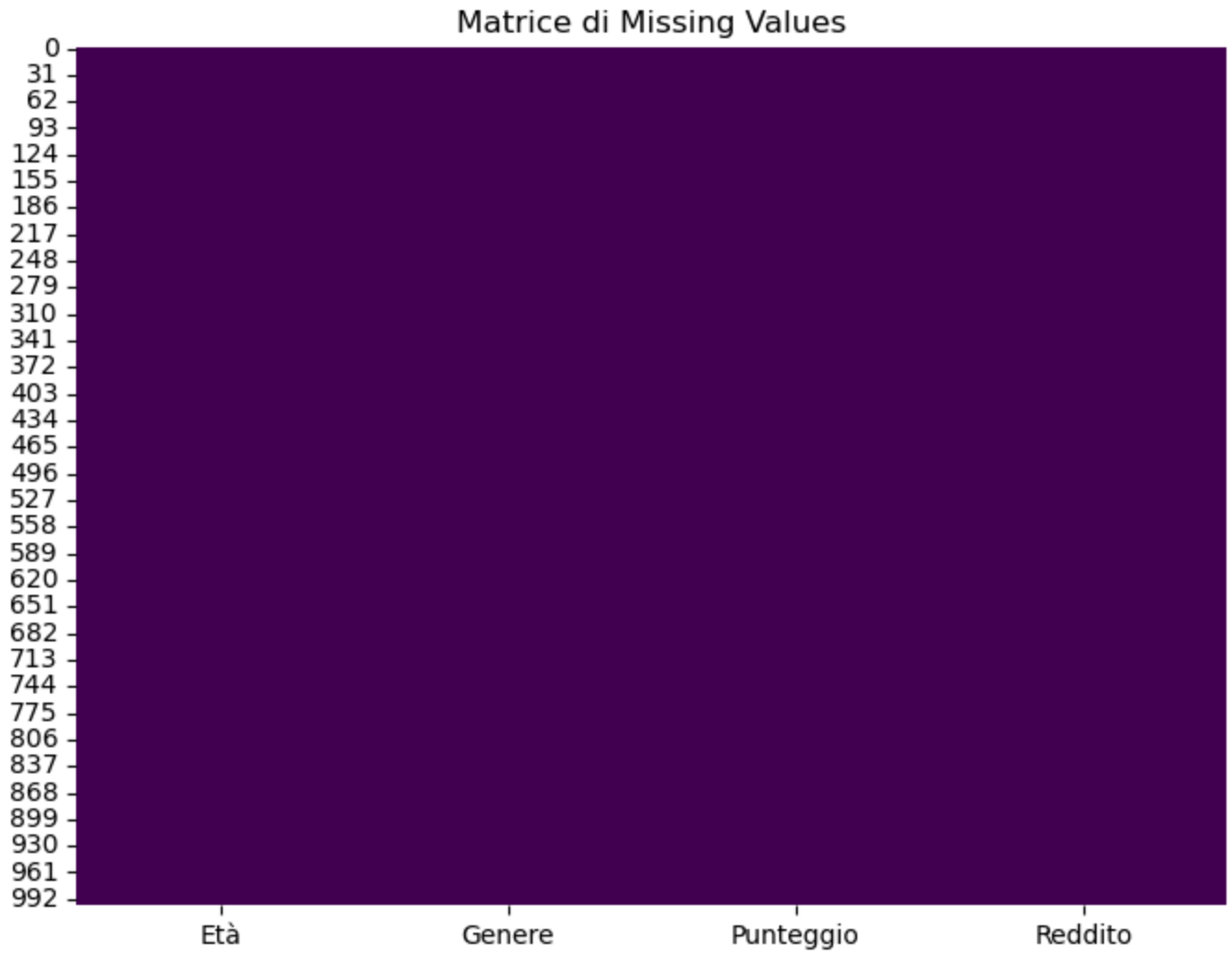
```
Valori mancanti per ciascuna colonna
Età          0
Genere       0
Punteggio    0
Reddito      0
dtype: int64
```

3.2 Matrice di Missing Values

In [23]:
```
# Visualizza una heatmap dei valori mancanti
# Imposta le dimensioni della figura
plt.figure(figsize=(8, 6))
 # Crea una heatmap per i valori mancanti nel DataFrame utilizzando Seaborn
sns.heatmap(df.isnull(), cmap='viridis', cbar=False,)
 titolo alla heatmap
```

```
plt.title('Matrice di Missing Values')
# Mostra la heatmap
plt.show
```

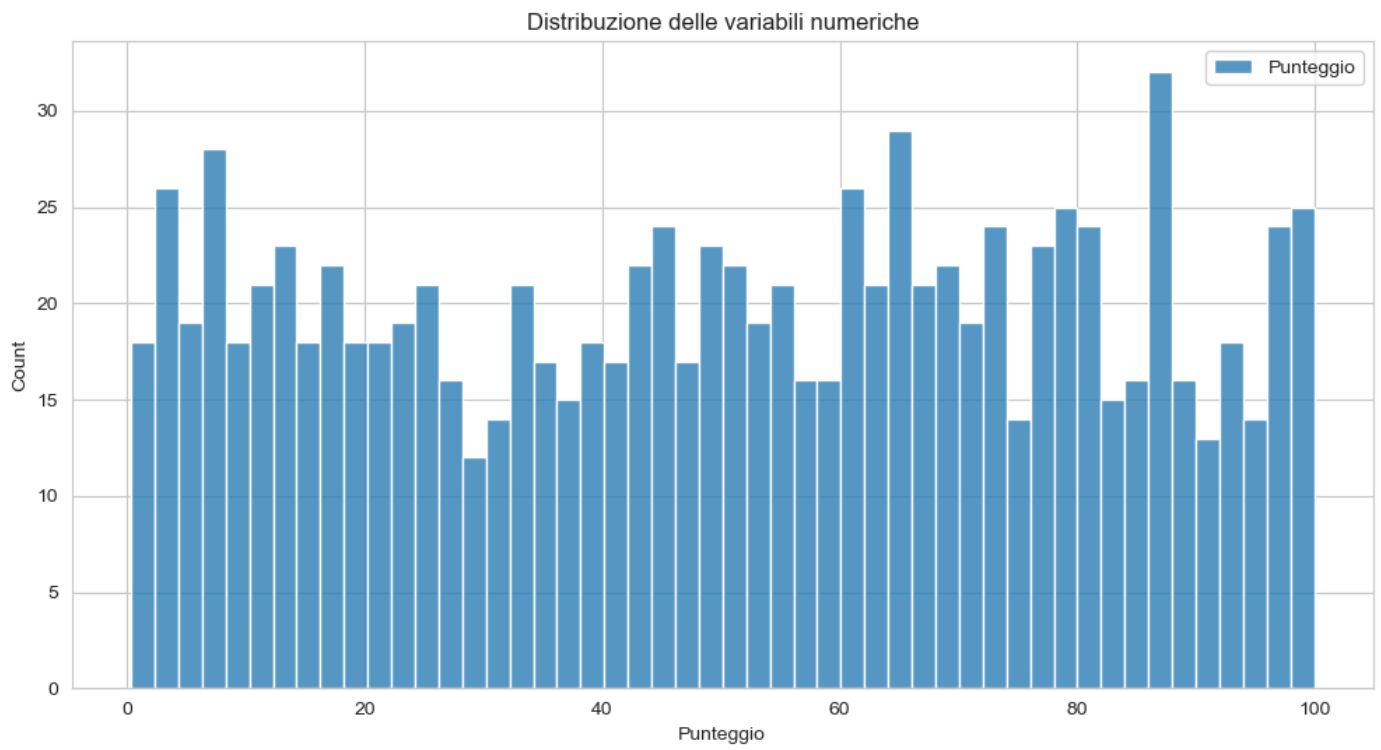
Out[23]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



3.3 Visualizzazione della Distribuzione della Variabile Numerica

In [24]:
```
# Visualizza la distribuzione delle variabili numeriche
# Imposta le dimensioni della figura
plt.figure(figsize=(12, 6))
# Imposta lo stile di sfondo del grafico
sns.set_style("whitegrid")
 # Crea un istogramma della variabile "Punteggio" senza KDE (Kernel Density Estimation)
sns.histplot(df["Punteggio"], kde=False, bins=50, label="Punteggio")
# Aggiunge la legenda al grafico
plt.legend()
# Aggiunge un titolo al grafico
plt.title('Distribuzione delle variabili numeriche')
# Mostra il grafico
plt.show
```

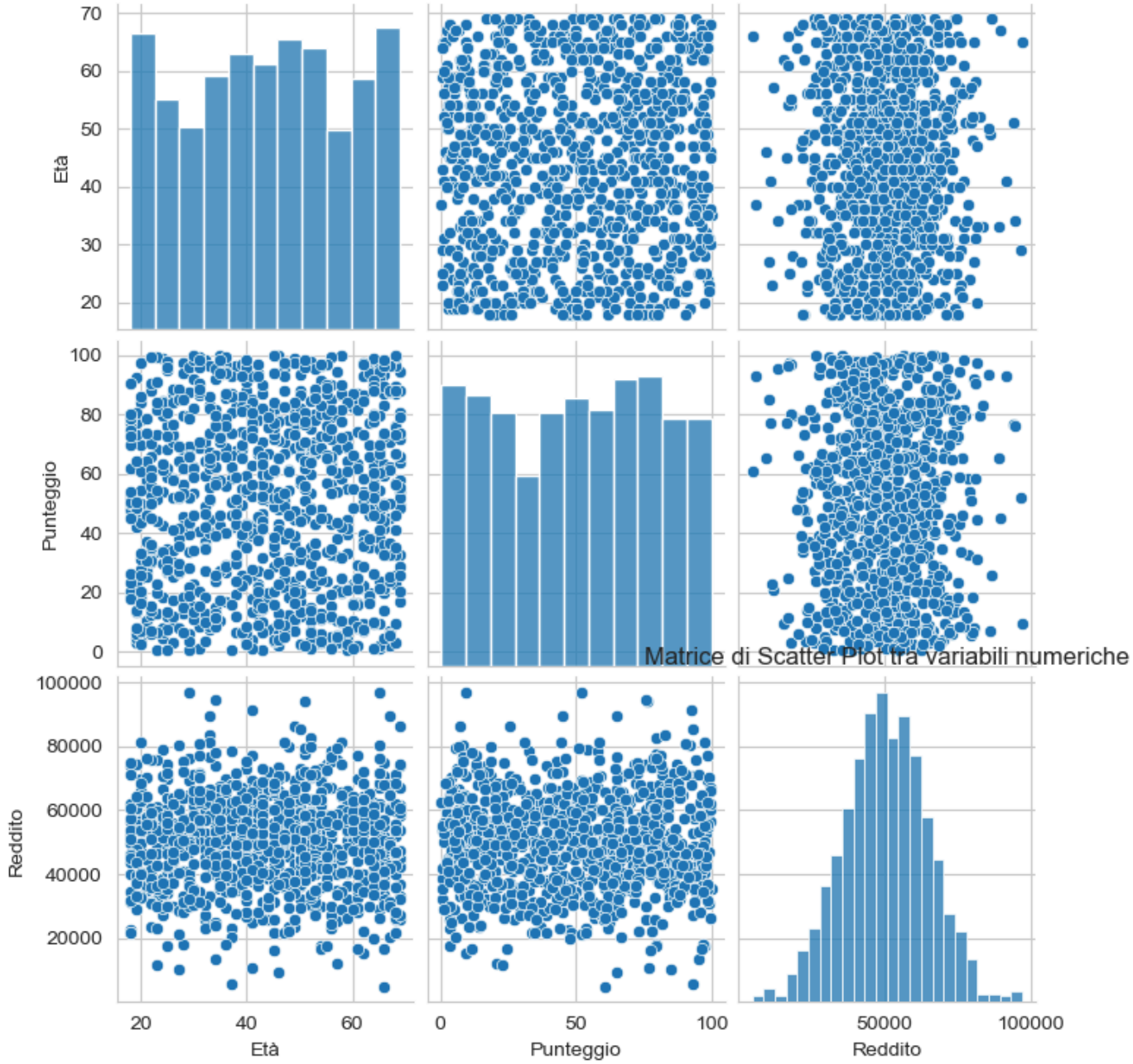
Out[24]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



3.4 Matrice di Scatter Plot tra Variabili Numeriche

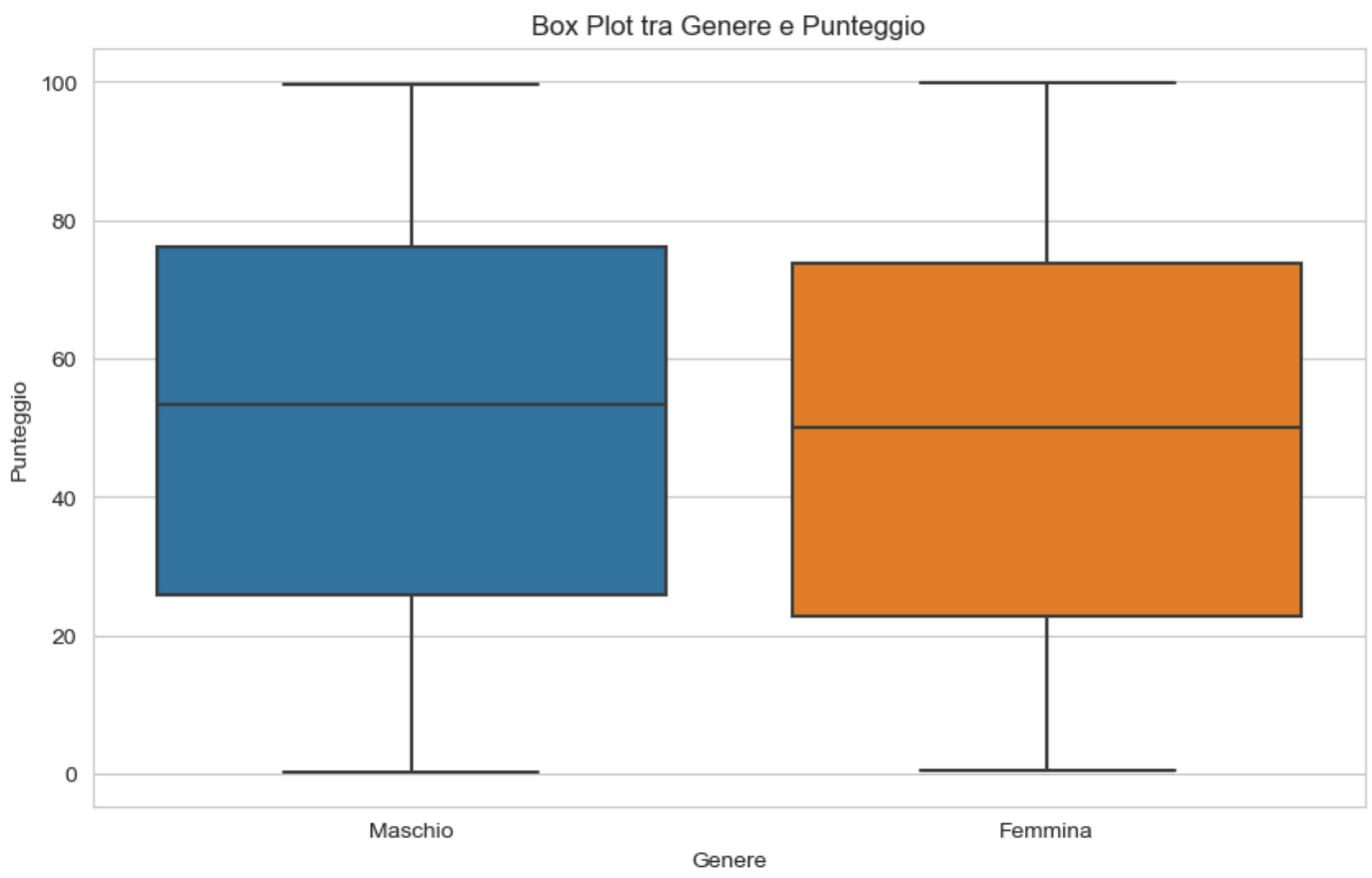
In [25]:
```
# Seleziona le colonne numeriche dal DataFrame
numeric_features = df.select_dtypes(include=[np.number])
# Crea una matrice di scatter plot tra le variabili numeriche
sns.pairplot(df[numeric_features.columns])
# Aggiunge un titolo al grafico
plt.title('Matrice di Scatter Plot tra variabili numeriche')
# Mostra il grafico
plt.show()
```

```
C:\Users\zetam\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The fig
ure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



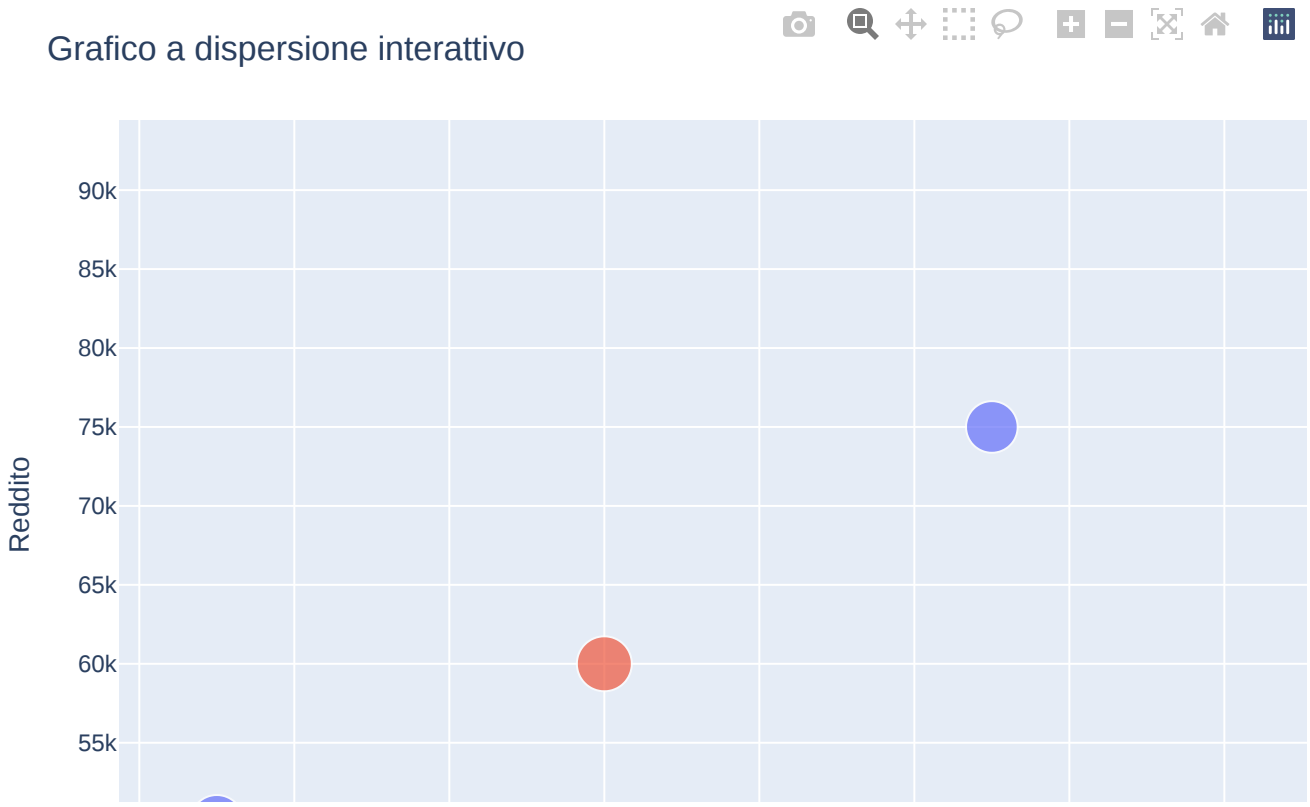
3.5 Box Plot della Distribuzione dei Punteggi per Genere

In [26]:
```
# Imposta le dimensioni della figura
plt.figure(figsize=(10, 6))
# Crea il boxplot
sns.boxplot(x='Genere', y='Punteggio', data=df)
# Aggiunge un titolo al grafico
plt.title('Box Plot tra Genere e Punteggio')
# Mostra il grafico
plt.show()
```



3.6 Creazione di un Grafico tra Età, Reddito e Punteggio

In [1]:
```
# Import delle librerie necessarie
# Per la manipolazione dei dati tramite DataFrame
import pandas as pd
# Per la creazione di grafici interattivi
import plotly.express as px

# Creazione di un dataframe di esempio
# Lista delle età
data = {'Età': [25, 30, 35, 40],
# Lista dei redditi
        'Reddito': [50000, 60000, 75000, 90000],
# Lista dei generi
        'Genere': ['Maschio', 'Femmina', 'Maschio', 'Femmina'],
# Lista dei punteggi
        'Punteggio': [80, 85, 75, 90]}
# Creazione del DataFrame utilizzando i dati forniti
df = pd.DataFrame(data)

# Creazione del grafico a dispersione
fig = px.scatter(df, x='Età', y='Reddito', color='Genere', size='Punteggio')
# Aggiornamento del layout del grafico per aggiungere un titolo
fig.update_layout(title='Grafico a dispersione interattivo')
# Mostra il grafico a dispersione interattivo
fig.show()
```



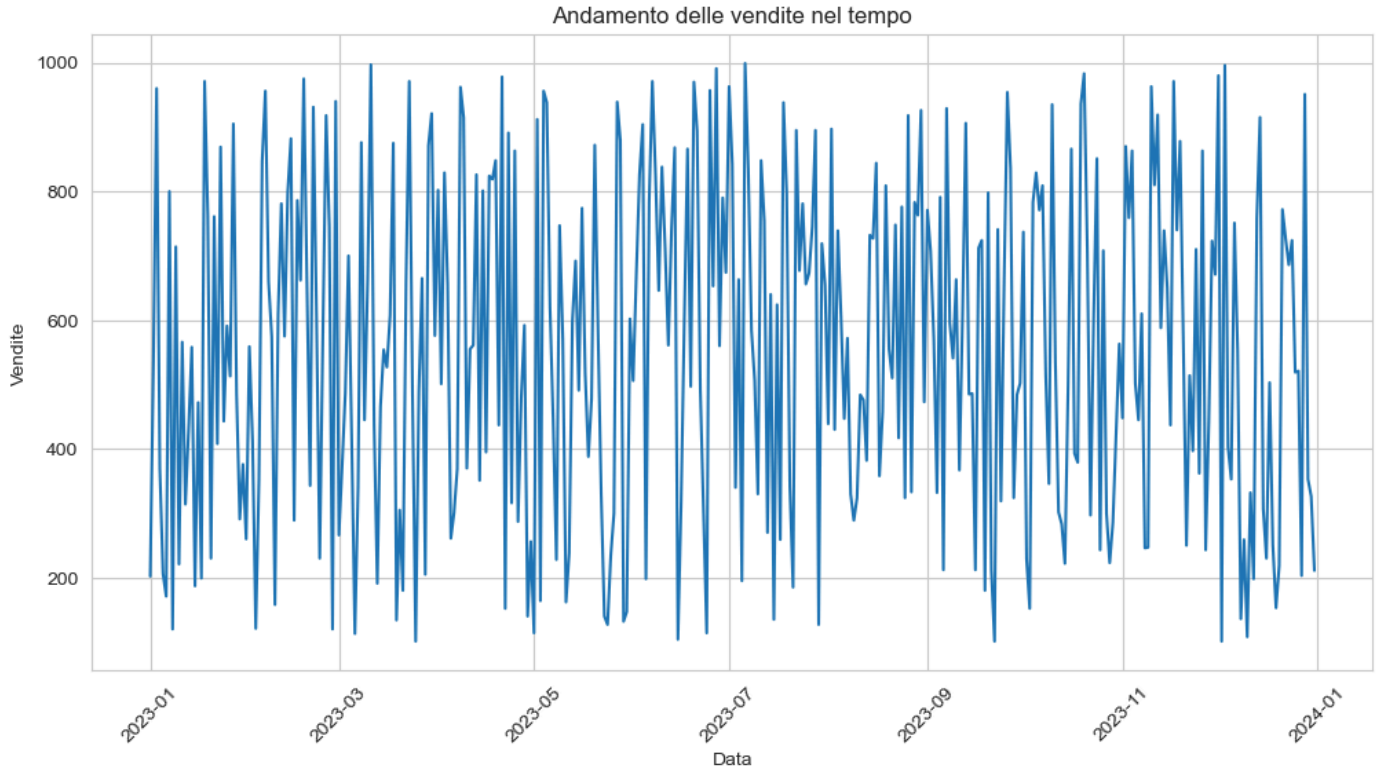
3.7 Analisi delle Vendite nel 2023 e Distribuzione delle Vendite per Prodotto

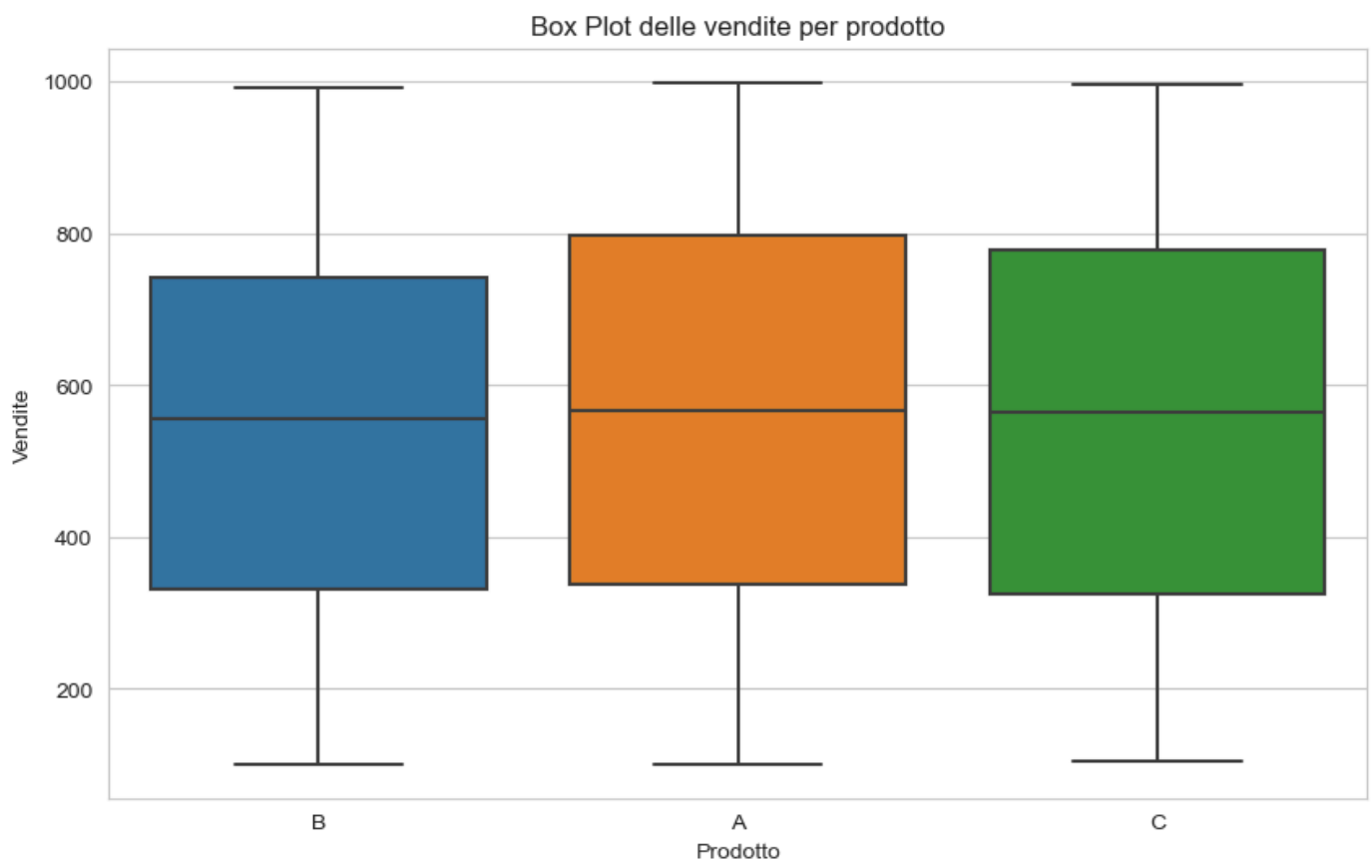
In [28]:

```
# Import delle librerie necessarie
import pandas as pd
# Per la generazione di dati casuali
import numpy as np
# Per la creazione di grafici
import matplotlib.pyplot as plt
# Per la creazione di visualizzazioni avanzate
import seaborn as sns

# Genera dati casuali per l'esplorazione
# Imposta il seed per la riproducibilità dei dati casuali
np.random.seed(42)
data = {
# Genera una serie di date giornaliere per tutto il 2023
    'Data': pd.date_range(start='2023-01-01', end='2023-12-31', freq='D'),
# Genera vendite casuali comprese tra 100 e 1000 per ogni giorno dell'anno
    'Vendite': np.random.randint(100, 1000, size=365),
# Assegna casualmente i prodotti A, B o C a ciascuna vendita
    'Prodotto': np.random.choice(['A', 'B', 'C'], size=365)
}
# Crea un DataFrame utilizzando i dati generati
df = pd.DataFrame(data)

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

```
# Visualizza un grafico delle vendite nel tempo
#  Imposta le dimensioni della figura
plt.figure(figsize=(12, 6))
# Crea un grafico a linea delle vendite nel tempo
sns.lineplot(x='Data', y='Vendite', data=df)
# Aggiunge un titolo al grafico
plt.title('Andamento delle vendite nel tempo')
# Aggiunge un'etichetta all'asse x
plt.xlabel('Data')
 # Aggiunge un'etichetta all'asse y
plt.ylabel('Vendite')
# Ruota le etichette sull'asse x per una migliore leggibilità
plt.xticks(rotation=45)
# Mostra il grafico delle vendite nel tempo
plt.show()

# Visualizza una box plot delle vendite per prodotto
# Imposta le dimensioni della figura
plt.figure(figsize=(10, 6))
# Crea un box plot delle vendite per prodotto
sns.boxplot(x='Prodotto', y='Vendite', data=df)
# Aggiunge un titolo al grafico
plt.title('Box Plot delle vendite per prodotto')
# Aggiunge un'etichetta all'asse x
plt.xlabel('Prodotto')
# Aggiunge un'etichetta all'asse y

plt.ylabel('Vendite')
# Mostra il box plot delle vendite per prodotto
plt.show()
```

```
        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
```





3.8 Creazione di un DataFrame con Dati Numerici e Categorici

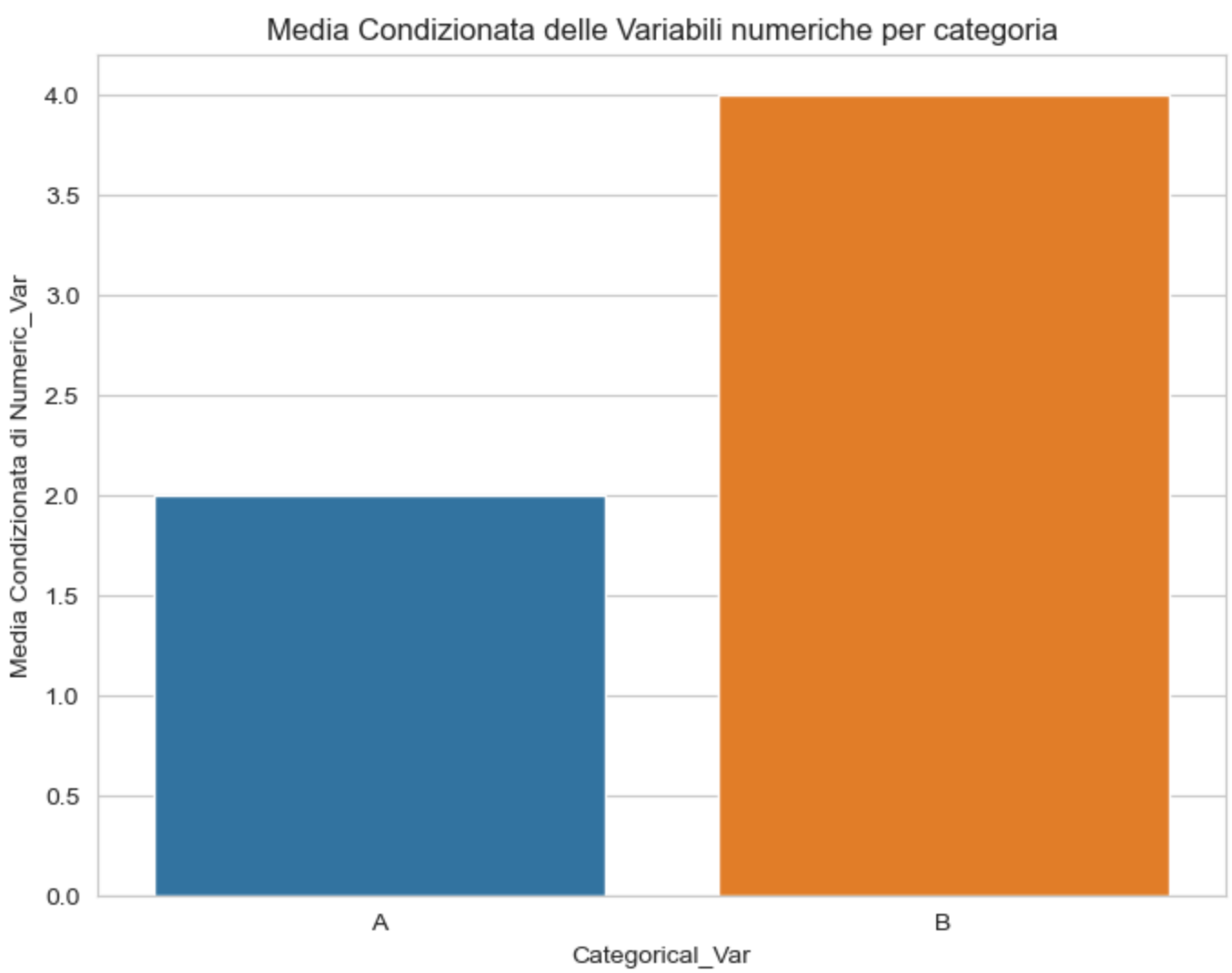
In [4]:
```python
# Import delle librerie necessarie

import pandas as pd
# Per la creazione di grafici
import matplotlib.pyplot as plt
# Per la gestione di array e dati numerici
import numpy as np
# Per la creazione di visualizzazioni statistiche
import seaborn as sns

# Genera dati di esempio
data = {
# Lista di valori numerici, inclusi valori mancanti
    'Numeric_Var': [1, 2, 3, 4, np.nan, 6],
# Lista di valori categorici

    'Categorical_Var': ['A', 'B', 'A', 'B', 'A', 'B']
}

# Crea un DataFrame
df = pd.DataFrame(data)
# Stampa il DataFrame per visualizzare i dati
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
```

3.9 Media Condizionata delle Variabili Numeriche per Categoria

In [30]:
```
#calcola la media condizionata
conditional_means = df['Numeric_Var'].fillna(df.groupby('Categorical_Var')['Numeric_Var'

#aggiorna la colonna numeric_var con la media condizionata
df['Numeric_Var'] = conditional_means
# Stampa il DataFrame per visualizzare i dati aggiornati
print(df)

#crea un graico a barre per mostrare la sedia condizionata per ogni categoria
# Imposta le dimensioni della figura
plt.figure(figsize=(8,6))
 # Crea un grafico a barre utilizzando Seaborn
sns.barplot(data=df, x='Categorical_Var', y='Numeric_Var', errorbar=None)
# Etichetta dell'asse x
plt.xlabel('Categorical_Var')
# Etichetta dell'asse y
plt.ylabel('Media Condizionata di Numeric_Var')
 # Titolo del grafico
plt.title('Media Condizionata delle Variabili numeriche per categoria')
# Mostra il grafico a barre della media condizionata 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
```

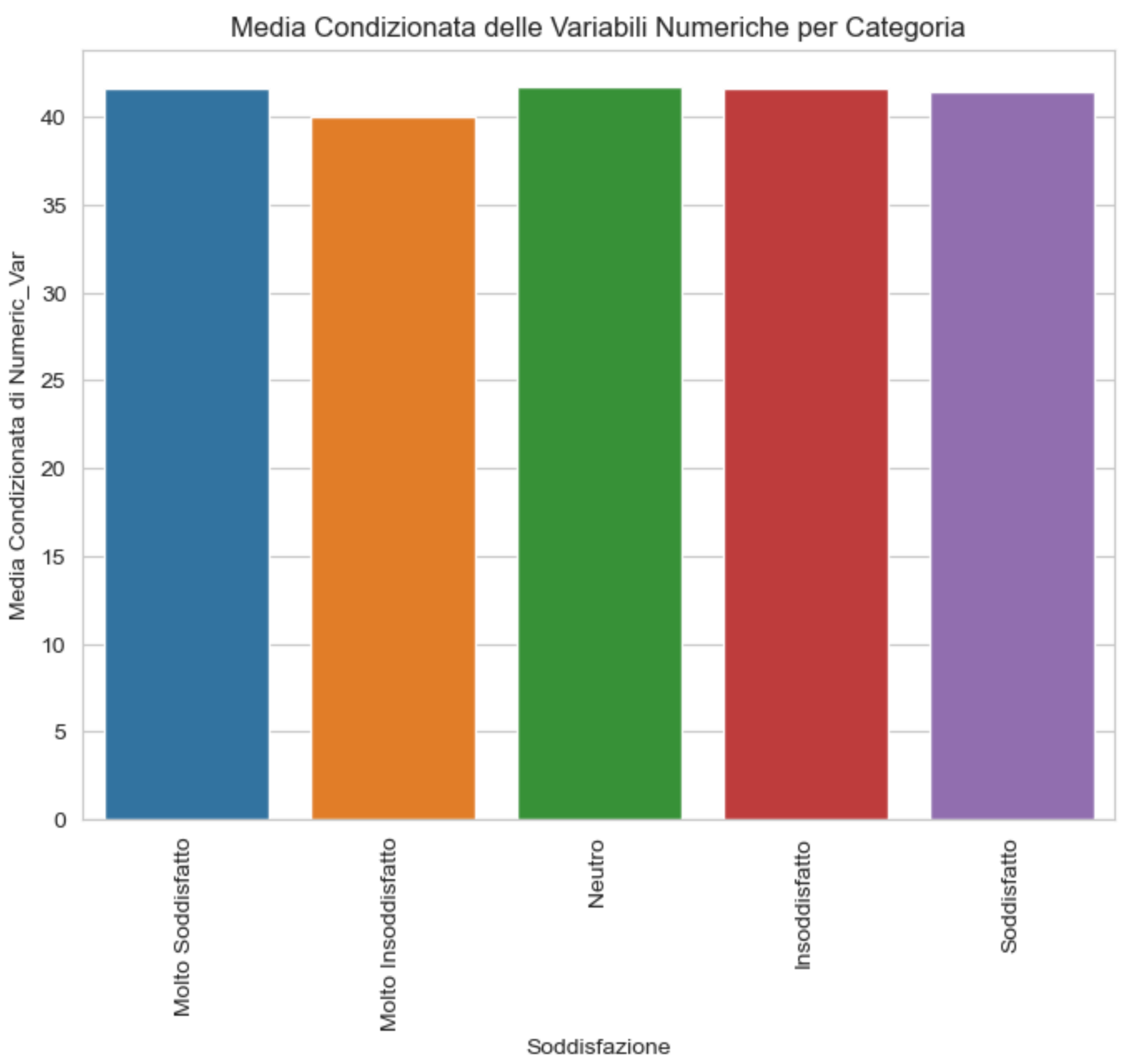


4.0 Analisi della Soddisfazione in Base all'Età con Media Condizionata delle Variabili Numeriche

In [31]:
```python
#Per la manipolazione dei dati tramite DataFram
import pandas as pd
 #  la generazione di dati casuali
import numpy as np
# la creazione di grafici
import matplotlib.pyplot as plt
# la creazione di visualizzazioni avanzate
import seaborn as sns

# Genera dati casuali per l'esplorazione
# Imposta il seed per la riproducibilità dei dati casuali
np.random.seed(42)
data = {
# Genera un array di età comprese tra 18 e 65 anni
    'Età': np.random.randint(18, 65, size=500),
# Genera un array di livelli di soddisfazione casuali
    'Soddisfazione': np.random.choice(['Molto Soddisfatto', 'Soddisfatto', 'Neutro', 'In
}
# Crea un DataFrame utilizzando i dati generati
df = pd.DataFrame(data)
# Stampa il DataFrame per visualizzare i dati
print(df)
# Calcola la media condizionata per l'età, raggruppata per livello di soddisfazione
conditional_means = df.groupby('Soddisfazione')['Età'].transform('mean')
# a nuova colonna 'Numeric_Var' al DataFrame con la media condizionata
```

```
df['Numeric_Var'] = conditional_means
# Stampa il DataFrame aggiornato per visualizzare la nuova colonna
print(df)

# Crea un grafico a barre per mostrare la media condizionata per ogni categoria
# Imposta le dimensioni della figura
plt.figure(figsize=(8, 6))
# Crea un grafico a barre utilizzando Seaborn
sns.barplot(data=df, x='Soddisfazione', y='Numeric_Var', ci=None)
# Etichetta dell'asse x
plt.xlabel('Soddisfazione')
# Etichetta dell'asse y
plt.ylabel('Media Condizionata di Numeric_Var')
# Titolo del grafico
plt.title('Media Condizionata delle Variabili Numeriche per Categoria')
# Ruota le etichette sull'asse x per una migliore leggibilità
plt.xticks(rotation=90)
# Mostra il grafico a barre della media condizionata per categoria di soddisfazione
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]
C:\Users\zetam\AppData\Local\Temp\ipykernel_3744\3910047455.py:22: FutureWarning:


The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.
```

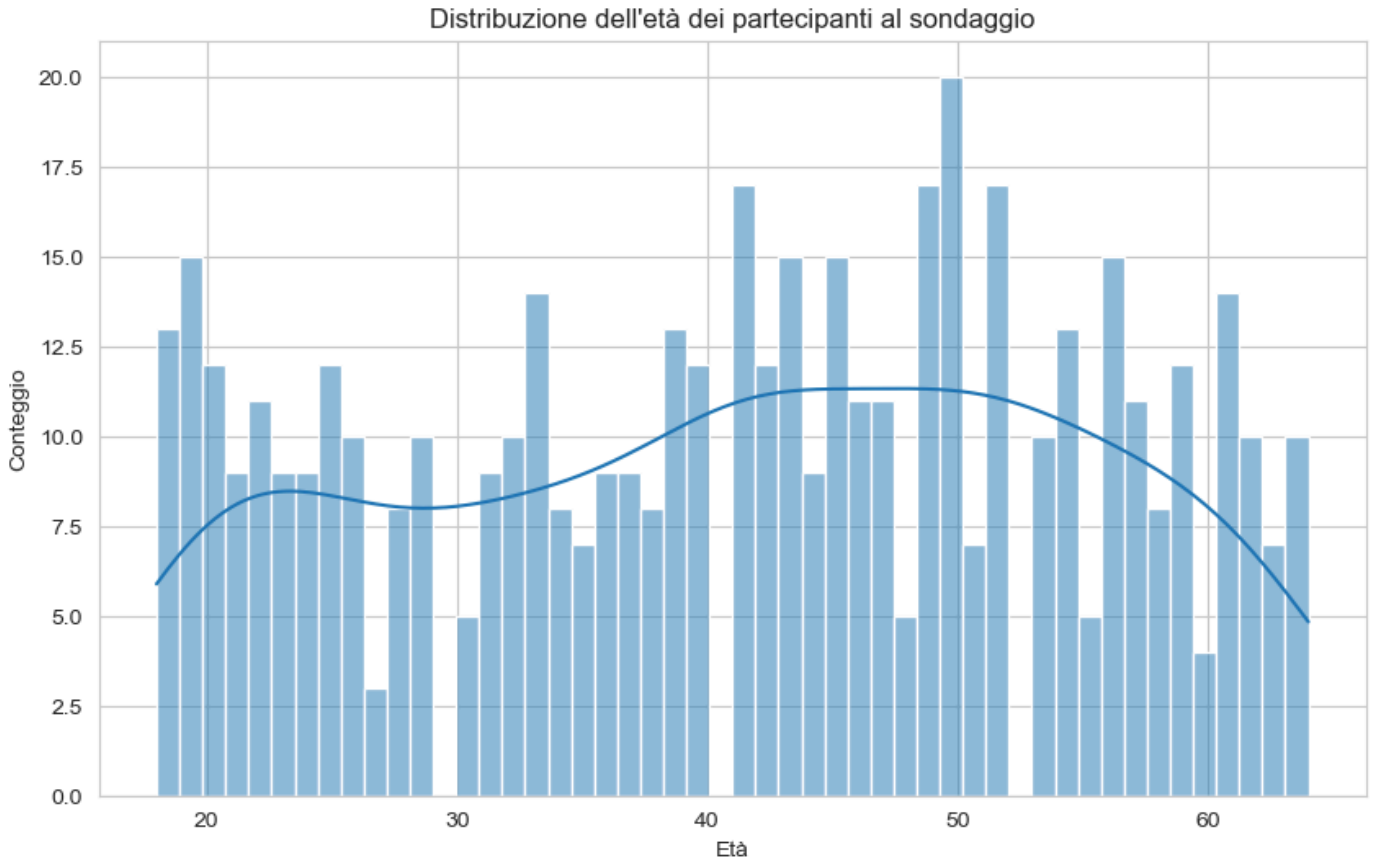


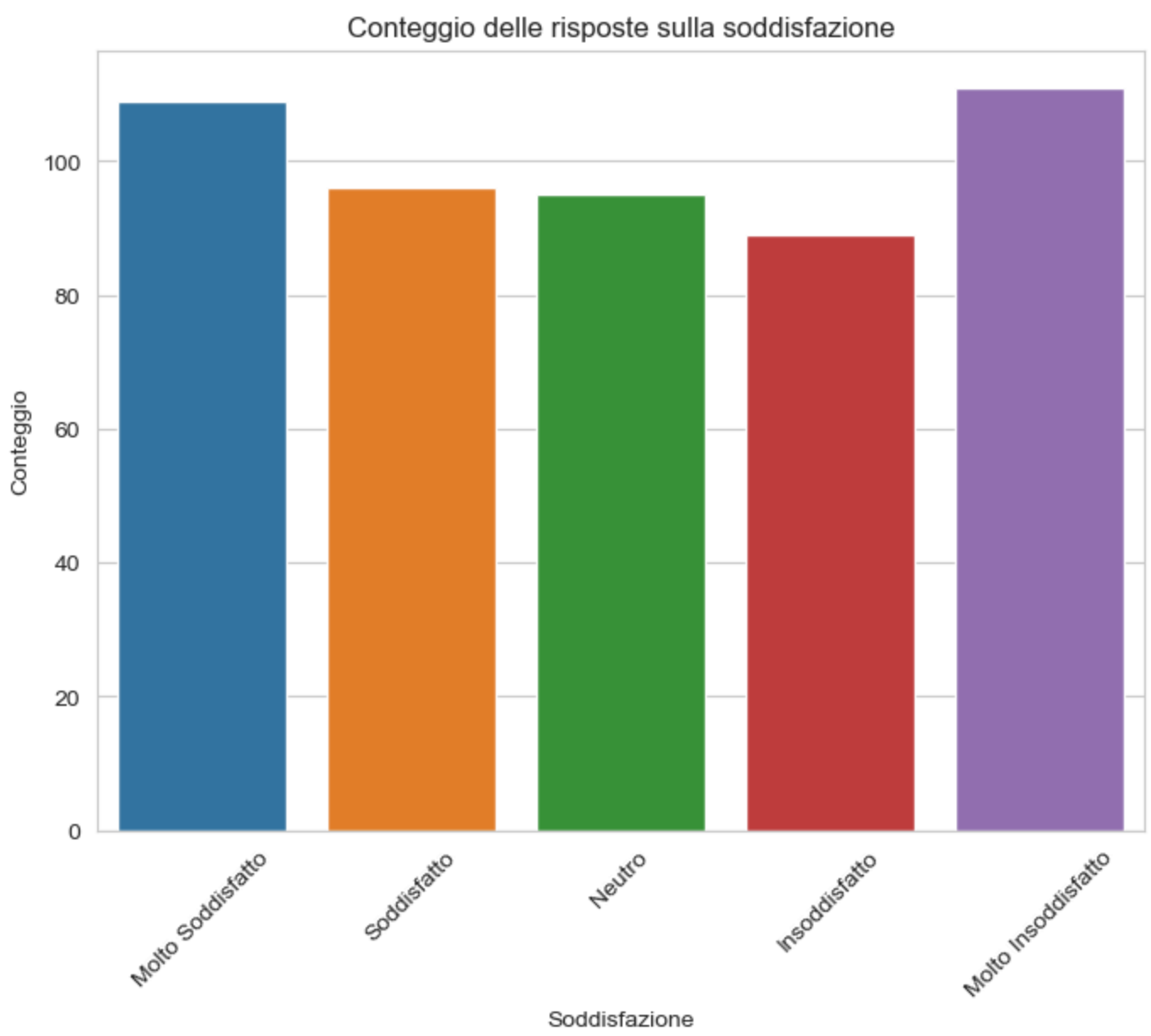
4.1 Visualizzazione delle Prime Righe del Dataset e Distribuzione dell'Età dei Partecipanti al Sondaggio

In [32]:
```
# Visualizza le prime righe del dataset
print(df.head())

# Visualizza una distribuzione dell'età
 # Imposta le dimensioni della figura
plt.figure(figsize=(10, 6))
# Crea un istogramma dell'età con KDE (Kernel Density Estimation)
sns.histplot(df['Età'], bins=50, kde=True)
# Aggiunge un titolo al grafico
plt.title('Distribuzione dell\'età dei partecipanti al sondaggio')
# Etichetta dell'asse x
plt.xlabel('Età')
# Etichetta dell'asse y
plt.ylabel('Conteggio')
# Mostra l'istogramma dell'età dei partecipanti al sondaggio
plt.show()

# Visualizza un conteggio delle risposte sulla soddisfazione
```

```
 # Imposta le dimensioni della figura
plt.figure(figsize=(8, 6))
# Crea un conteggio delle risposte sulla soddisfazione
sns.countplot(x='Soddisfazione', data=df, order=['Molto Soddisfatto', 'Soddisfatto', 'Ne
# Aggiunge un titolo al grafico
plt.title('Conteggio delle risposte sulla soddisfazione')
 # Etichetta dell'asse x
plt.xlabel('Soddisfazione')
# Etichetta dell'asse y
plt.ylabel('Conteggio')
 # Ruota le etichette sull'asse x per una migliore leggibilità
plt.xticks(rotation=45)
# Mostra il grafico a barre del conteggio delle risposte sulla soddisfazione
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

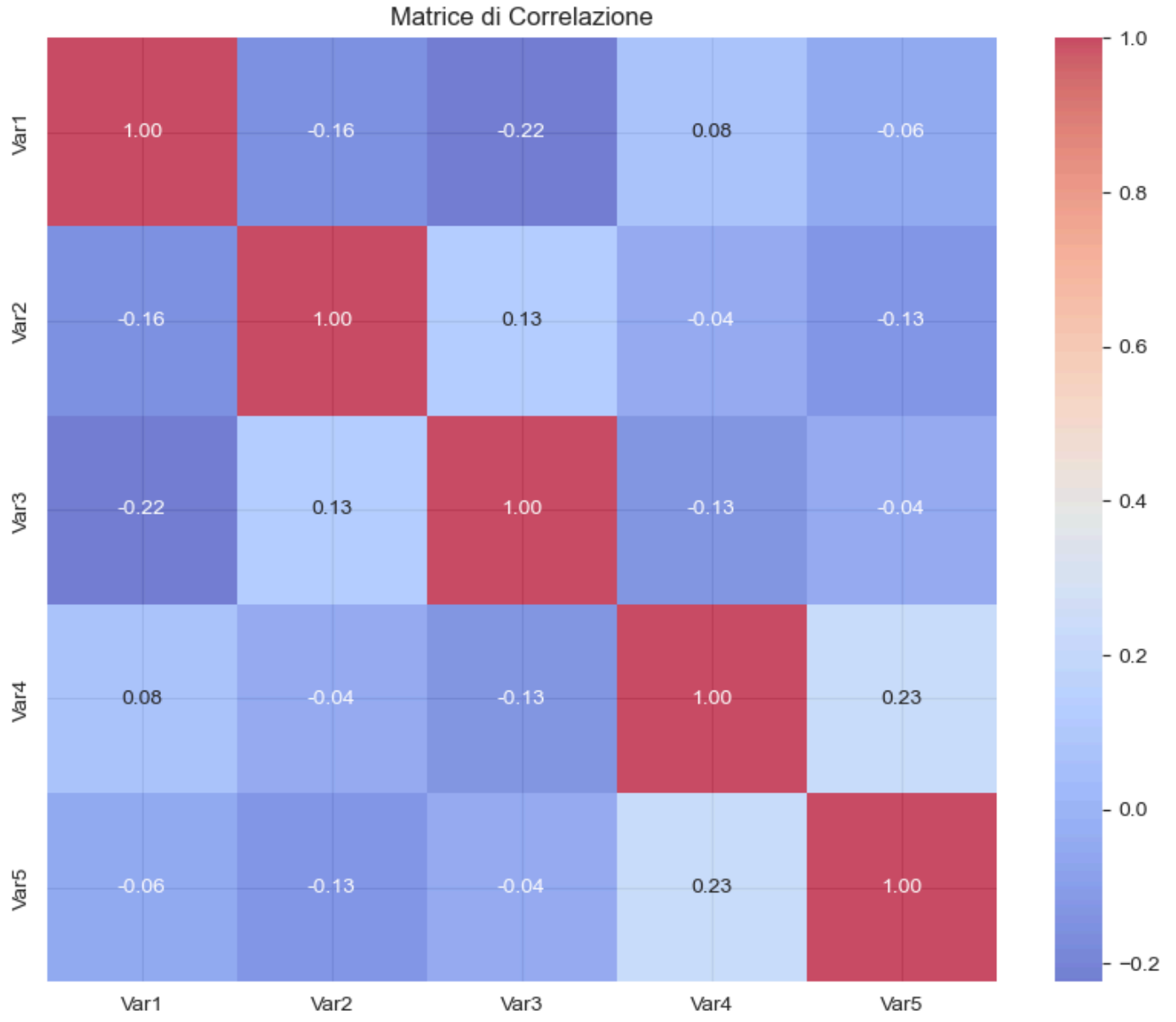




4.2 Visualizzazione della Matrice di Correlazione tra Variabili Numeriche

In [35]:
```
# Per la generazione di dati casuali
import numpy as np
# Per la manipolazione dei dati tramite DataFrame
import pandas as pd
# Per la creazione di visualizzazioni statistiche
import seaborn as sns
# Per la creazione di grafici
import matplotlib.pyplot as plt

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

# 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)

# Seleziona solo le colonne numeriche per il calcolo della correlazione
= data.select_dtypes(include=[np.number])
```

```
# Calcola la matrice di correlazione tra tutte le variabili numeriche
correlation_matrix = numeric_data.corr()

# Visualizza la matrice di correlazione come heatmap
# Imposta le dimensioni della figura
plt.figure(figsize=(10, 8))
# Crea una heatmap della matrice di correlazione con annotazioni e colormap 'coolwarm'
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", alpha=0.7)
# Aggiunge un titolo alla heatmap
plt.title("Matrice di Correlazione")
# Mostra la heatmap della matrice di correlazione
plt.show()
```



4.3 Creazione di un DataFrame con Dati Casuali e Valori Mancanti

In [36]:
```
# Per la manipolazione dei dati tramite DataFrame
import pandas as pd
# Per la generazione di dati casuali
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
# Numero di righe nel DataFrame
n_rows = 10000
# Colonna di categorie casuali
df['CatCol1'] = np.random.choice(['A', 'B', 'C'], size=n_rows)
# Altra colonna di categorie casuali
df['CatCol2'] = np.random.choice(['X', 'Y'], size=n_rows)
 # Colonna di valori casuali con distribuzione normale
df['NumCol1'] = np.random.randn(n_rows)
# Colonna di valori casuali interi tra 1 e 100
df['NumCol2'] = np.random.randint(1, 100, size=n_rows)
# Colonna di valori casuali con distribuzione uniforme tra 0 e 1
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:
# Numero casuale di valori mancanti per ciascuna colonna
    num_missing_values = np.random.randint(0, total_missing_values + 1)
# Indici casuali per i valori mancanti
    missing_indices = np.random.choice(n_rows, size=num_missing_values, replace=False)
# Imposta i valori mancanti nei rispettivi indici e colonne
    df.loc[missing_indices, column] = np.nan
# Visualizza il DataFrame con i dati casuali e i valori mancanti
df
```

Out[36]:

	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 × 5 columns

4.4 Conteggio delle Righe con Dati Mancanti nel DataFrame

In [37]:
```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
len(righe_con_dati_mancanti)
```

Out[37]:
```
3648
```

4.5 Righe con Dati Mancanti

In [38]:
```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

Out[38]:

	CatCol1	CatCol2	NumCol1	NumCol2	NumCol3
0	A	NaN	0.440877	49.0	0.246007
5	B	NaN	NaN	71.0	0.752397
6	B	X	0.080686	31.0	NaN
8	B	Y	-1.291483	NaN	0.868791
12	C	Y	-1.193705	8.0	NaN
...	...	...	...	...	...
9986	C	X	-0.909994	NaN	0.767918
9988	A	Y	NaN	35.0	0.149513
9989	A	NaN	-0.148047	NaN	0.326089
9992	A	Y	-0.048300	58.0	NaN
9998	C	Y	0.004278	NaN	NaN

3648 rows × 5 columns

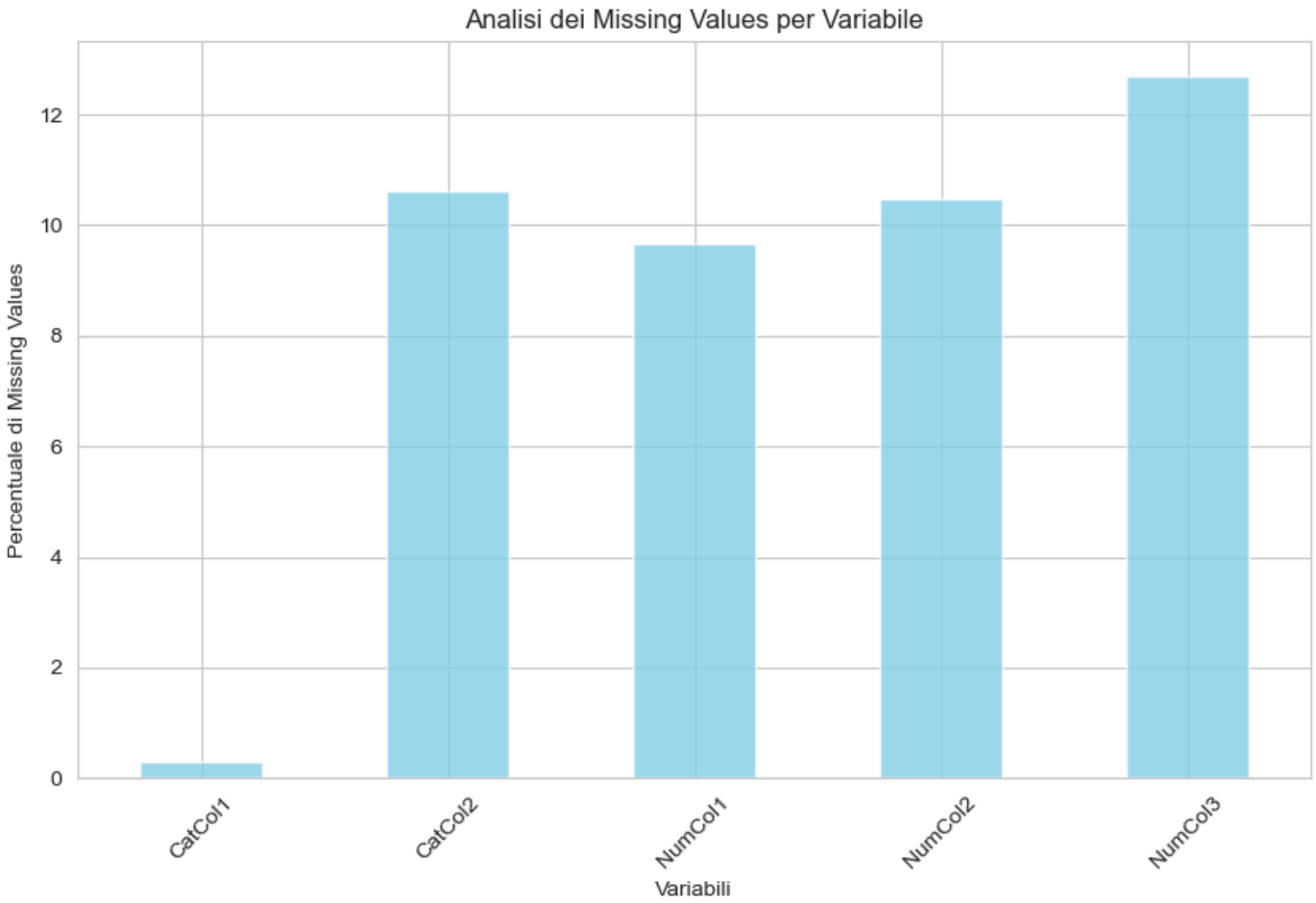
4.6 Calcolo della Percentuale di Valori Mancanti per Variabile

In [39]:
```
missing_percent = (df.isnull().sum() / len(df)) * 100
missing_percent
```

Out[39]:
```
CatCol1     0.29
CatCol2    10.63
NumCol1     9.67
NumCol2    10.48
NumCol3    12.69
dtype: float64
```

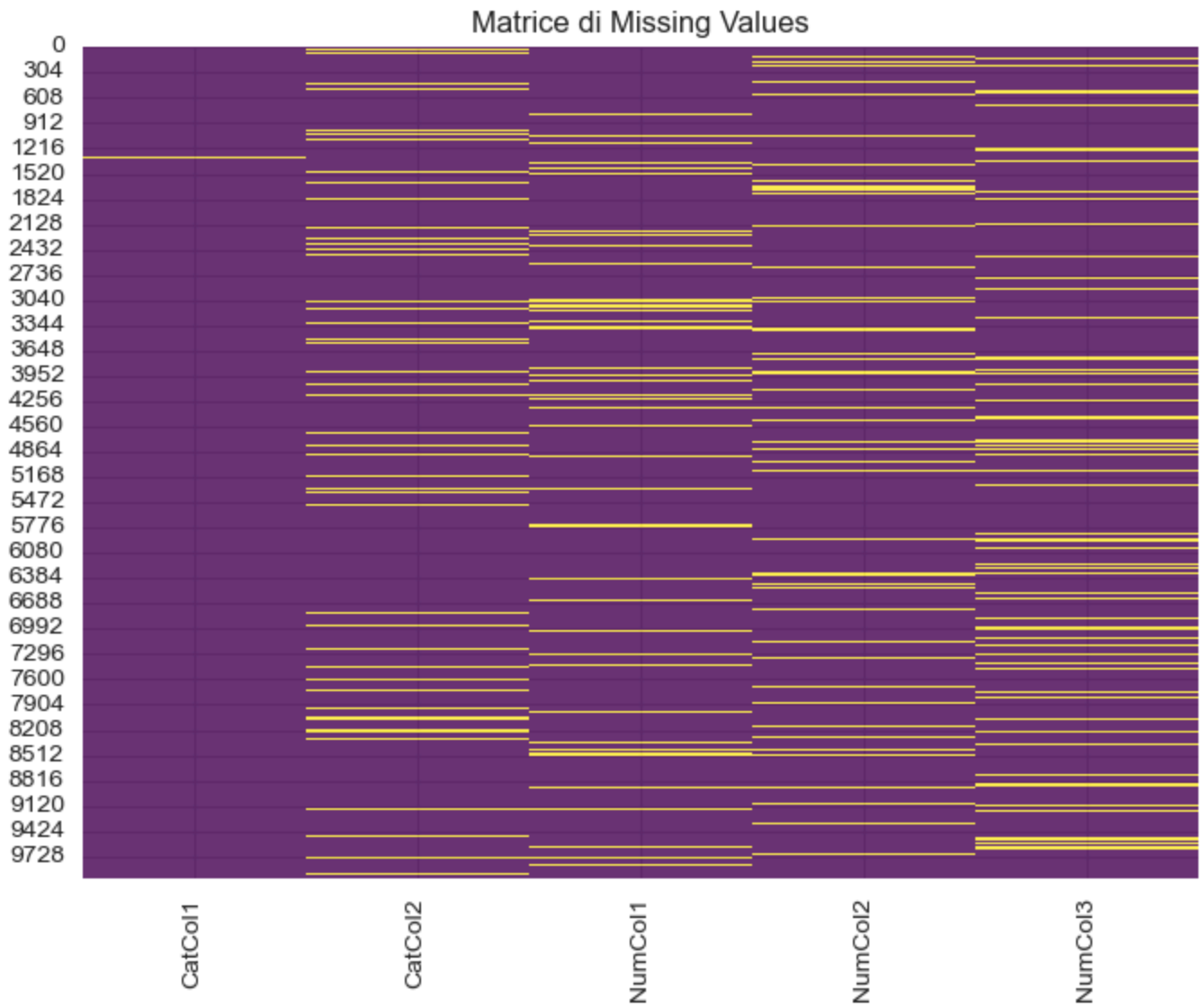
4.7 Analisi dei Missing Values per Variabile tramite Grafico a Barre

In [40]:
```
# Crea il grafico a
# Imposta le dimensioni della figura
plt.figure(figsize=(10, 6))
 # Crea un grafico a barre utilizzando Pandas
missing_percent.plot(kind='bar', color='skyblue',alpha=0.8)
# Etichetta dell'asse x
plt.xlabel('Variabili')
# Etichetta dell'asse y
plt.ylabel('Percentuale di Missing Values')
 # Titolo del grafico
plt.title('Analisi dei Missing Values per Variabile')
# Ruota le etichette sull'asse x per una migliore leggibilità
plt.xticks(rotation=45)
# Mostra il grafico a barre
plt.show()
```



4.8 Visualizzazione della Matrice di Missing Values tramite Heatmap

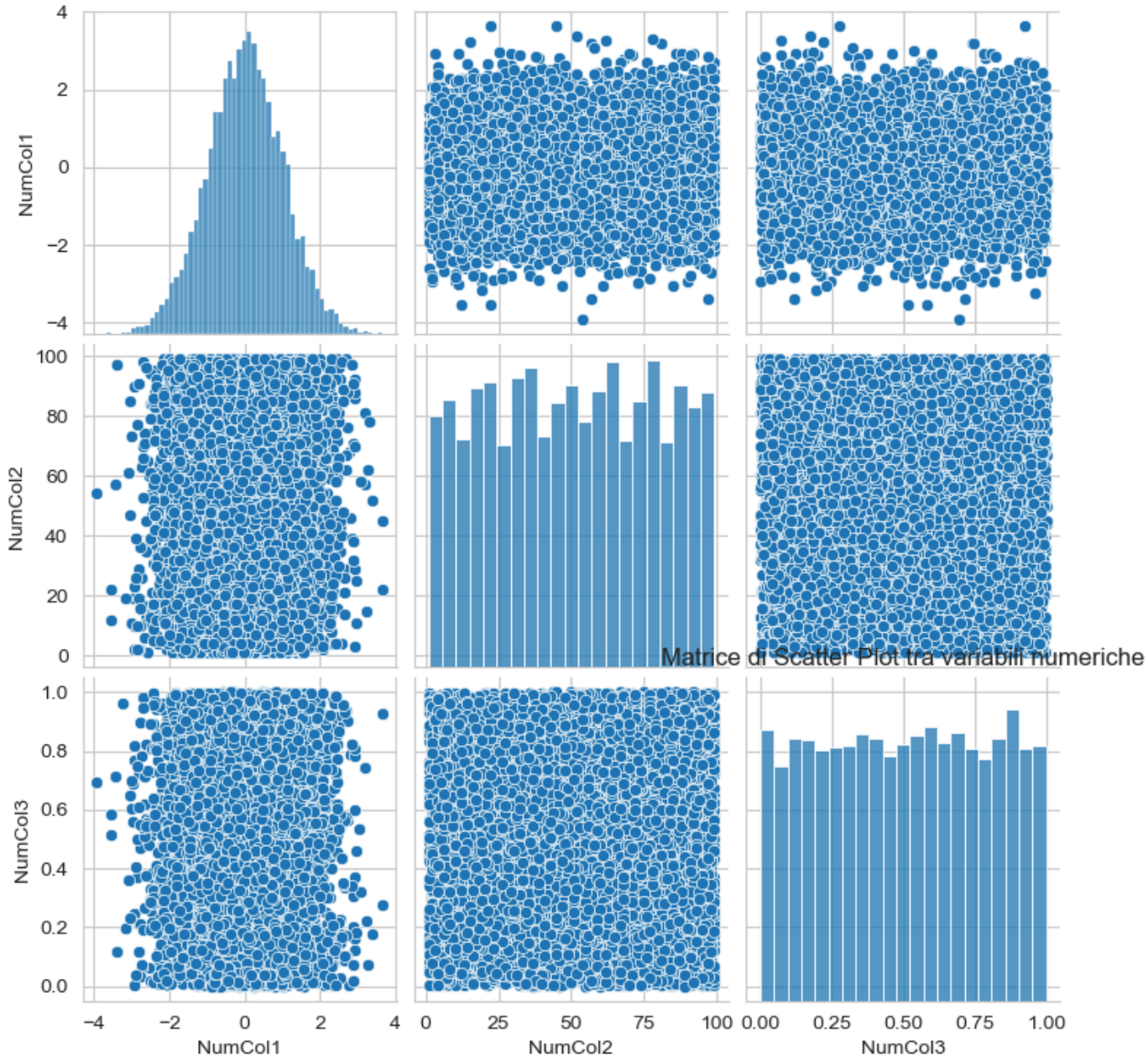
In [41]:
```
# Calcola una matrice booleana che mostra la presenza di valori mancanti in ogni cella d
missing_matrix = df.isnull()
# Crea una heatmap per visualizzare la matrice di valori mancanti
# Imposta le dimensioni della figura
plt.figure(figsize=(8, 6))
# Crea una heatmap utilizzando Seaborn
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
# Aggiunge un titolo alla heatmap
plt.title('Matrice di Missing Values')
 # Ruota le etichette sull'asse x per una migliore leggibilità
plt.xticks(rotation=90)
# Mostra la heatmap della matrice di valori mancanti
plt.show()
```



4.9 Matrice di Scatter Plot tra Variabili Numeriche

In [42]:
```
# Seleziona solo le colonne numeriche dal DataFrame
numeric_features = df.select_dtypes(include=[np.number])
# Crea un pairplot per mostrare la matrice di scatter plot tra le variabili numeriche
 # Crea un pairplot utilizzando Seaborn
sns.pairplot(df[numeric_features.columns])
# Aggiunge un titolo al pairplot
plt.title('Matrice di Scatter Plot tra variabili numeriche')
# Mostra il pairplot
plt.show()
```

```
C:\Users\zetam\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```



5.0 Rimozione delle Righe con Valori Mancanti nelle Colonne CatCol1 e CatCol2

In [43]:
```
# Rimuove le righe che hanno valori mancanti nelle colonne "CatCol1" e "CatCol2"
df = df.dropna(subset=["CatCol1", "CatCol2"], how='all')
```

Out[43]:

	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

9995 rows × 5 columns

5.1 Eliminazione di Righe con Valori NaN in Specifiche Colonne di un DataFrame

In [44]:
```
df = df.dropna(subset=["NumCol1", "NumCol2", "NumCol3"], how='all')
df
```

Out[44]:

	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

9975 rows × 5 columns

5.2 Riempimento di Valori Mancanti in Colonne Numeriche e Categoriche

In [45]:
```
numeric_cols = df.select_dtypes(include=['number'])
categorical_cols = df.select_dtypes(exclude=['number'])
df.loc[:, categorical_cols.columns] = df[categorical_cols.columns].fillna(df[categorical
conditional_means = df[categorical_cols.columns].fillna(df.groupby('CatCol1')[numeric_co
```

```
df.loc[:, numeric_cols.columns] = conditional_means
print(df)
```

```
     CatCol1 CatCol2  NumCol1  NumCol2  NumCol3
0          A       Y      NaN      NaN      NaN
1          A       Y      NaN      NaN      NaN
2          C       X      NaN      NaN      NaN
3          A       Y      NaN      NaN      NaN
4          B       X      NaN      NaN      NaN
...      ...     ...      ...      ...      ...
9995       C       Y      NaN      NaN      NaN
9996       C       Y      NaN      NaN      NaN
9997       A       X      NaN      NaN      NaN
9998       C       Y      NaN      NaN      NaN
9999       A       X      NaN      NaN      NaN

[9975 rows x 5 columns]
```

Import Data

1.0 Caricamento e Visualizzazione dei Dati dei Pokemon

In [46]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

percorso_file_csv = "C:\\Users\\zetam\\Desktop\\2 Superiore\\Robotica\\pokemon\\pokemons
df = pd.read_csv(percorso_file_csv)
print (df.head())
```

```
   id        name      rank    generation evolves_from  type1   type2  hp  \
0   1   bulbasaur  ordinary  generation-i      nothing  grass  poison  45
1   2     ivysaur  ordinary  generation-i    bulbasaur  grass  poison  60
2   3    venusaur  ordinary  generation-i      ivysaur  grass  poison  80
3   4  charmander  ordinary  generation-i      nothing   fire     NaN  39
4   5  charmeleon  ordinary  generation-i   charmander   fire     NaN  58

   atk  def  spatk  spdef  speed  total  height  weight  \
0   49   49     65     65     45    318       7      69
1   62   63     80     80     60    405      10     130
2   82   83    100    100     80    525      20    1000
3   52   43     60     50     65    309       6      85
4   64   58     80     65     80    405      11     190

              abilities                                               desc
0  overgrow chlorophyll   A strange seed was planted on its back at birt...
1  overgrow chlorophyll   When the bulb on its back grows large, it appe...
2  overgrow chlorophyll   The plant blooms when it is absorbing solar en...
3     blaze solar-power   Obviously prefers hot places. When it rains, s...
4     blaze solar-power   When it swings its burning tail, it elevates t...
```

1.1 Caricamento dei Dati della Serie A dal File Excel

In [47]:
```
import pandas as pd
percorso_file_excel = "C:\\Users\\zetam\\Desktop\\2 Superiore\\Robotica\\serieA.xlsx"
df = pd.read_excel(percorso_file_excel, sheet_name='10-11')
df
```

Out[47]:

	position	team	Pt	Played	Won	Net	lose	Goals made	Goals suffered	Difference goals
0	1	Milan Milan	82	38	24	10	4	65	24	41
1	2	Inter Inter	76	38	23	7	8	69	42	27
2	3	Napoli Napoli	70	38	21	7	10	59	39	20
3	4	Udinese Udinese	66	38	20	6	12	65	43	22
4	5	Lazio Lazio	66	38	20	6	12	55	39	16
5	6	Roma Roma	63	38	18	9	11	59	52	7
6	7	Juventus Juventus	58	38	15	13	10	57	47	10
7	8	Palermo Palermo	56	38	17	5	16	58	63	-5
8	9	Fiorentina Fiorentina	51	38	12	15	11	49	44	5
9	10	Genoa Genoa	51	38	14	9	15	45	47	-2
10	11	Chievo Chievo	46	38	11	13	14	38	40	-2
11	12	Parma Parma	46	38	11	13	14	39	47	-8
12	13	Catania Catania	46	38	12	10	16	40	52	-12
13	14	Cagliari Cagliari	45	38	12	9	17	44	51	-7
14	15	Cesena Cesena	43	38	11	10	17	38	50	-12
15	16	Bologna Bologna (-3)	42	38	11	12	15	35	52	-17
16	17	Lecce Lecce	41	38	11	8	19	46	66	-20
17	18	Sampdoria Sampdoria	36	38	8	12	18	33	49	-16
18	19	Brescia Brescia	32	38	7	11	20	34	52	-18
19	20	Bari Bari	24	38	5	9	24	27	56	-29

1.2 Aggregazione di File CSV in una Lista di DataFrame

In [48]:
```
import os
import pandas as pd
percorso_cartella = 'C:\\Users\\zetam\\Desktop\\2 Superiore\\Robotica\\serieAnuovo'
lista_dataframes = []

for nome_file in os.listdir(percorso_cartella):
    if nome_file.endswith(".csv"):
        percorso_file_csv = os.path.join(percorso_cartella, nome_file)
        df = pd.read_csv(percorso_file_csv)
        lista_dataframes.append(df)
```

1.3 Accesso ai Dati di un DataFrame da una Lista

In [49]:
```
lista_dataframes[23]
```

Out[49]:

	Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	...	BbAv<2.5	BbAH	BbAH
0	I1	20/08/16	Juventus	Fiorentina	2	1	H	1.0	0.0	H	...	1.78	36	-1.0
1	I1	20/08/16	Roma	Udinese	4	0	H	0.0	0.0	D	...	2.04	32	-1.5
2	I1	21/08/16	Atalanta	Lazio	3	4	A	0.0	3.0	A	...	1.63	31	0.2
3	I1	21/08/16	Bologna	Crotone	1	0	H	0.0	0.0	D	...	1.53	31	-0.5
4	I1	21/08/16	Chievo	Inter	2	0	H	0.0	0.0	D	...	1.60	31	0.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
375	I1	28/05/17	Inter	Udinese	5	2	H	3.0	0.0	H	...	3.39	19	-1.5
376	I1	28/05/17	Palermo	Empoli	2	1	H	0.0	0.0	D	...	2.17	19	1.0
377	I1	28/05/17	Roma	Genoa	3	2	H	1.0	1.0	D	...	4.67	15	-3.0
378	I1	28/05/17	Sampdoria	Napoli	2	4	A	0.0	2.0	A	...	4.32	15	2.2
379	I1	28/05/17	Torino	Sassuolo	5	3	H	3.0	2.0	H	...	3.62	19	-1.0

380 rows × 64 columns

In [ ]: