I Missing Values

Parte dei missing values delle lezioni

Importazione Delle Librerie

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

Creazione e Visualizzazione del DataFrame

```
# Creazione del DataFrame con Possibili Valori Mancanti
import pandas as pd

# Definizione del dataset con potenziali valori mancanti
dataset = [
    {"età": 25, "punteggio": 90, "ammesso": 1},
    {"età": None, "punteggio": 85, "ammesso": 0},  # None indica un
valore mancante
    {"età": 28, "punteggio": None, "ammesso": 1},
    {"età": None, "punteggio": 75, "ammesso": 1},
    {"età": 23, "punteggio": None, "ammesso": None},
    {"età": 23, "punteggio": 77, "ammesso": None},
]

# Conversione del dataset in un DataFrame pandas
df = pd.DataFrame(dataset)

# Visualizzazione del DataFrame
print(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
```

Accesso a Colonne Specifiche

```
# Accesso alla colonna 'punteggio' del DataFrame
df["punteggio"]
```

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

Accesso a Dati Demografici

```
# Accesso alla colonna 'età' del DataFrame
df["età"]
```

```
0    25.0
1     NaN
2    28.0
3     NaN
4    23.0
5    23.0
Name: età, dtype: float64
```

Identificazione Valori Mancanti

```
# Identificazione delle righe con dati mancanti nel DataFrame
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
```

Calcolo Dati Mancanti

```
# Calcolo del numero totale di dati mancanti nel DataFrame
totale_dati_mancanti = righe_con_dati_mancanti.shape[0]
totale_dati_mancanti
```

```
5
```

Visualizzazione Dati Mancanti

```
# Visualizzazione delle righe con dati mancanti e del 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
```

Creazione DataFrame con Dati Mancanti

```
# Creazione di un DataFrame 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"},
]

# Conversione del dataset in 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
```

Rimozione Dati Mancanti

```
# Rimozione delle righe con dati mancanti dal DataFrame
df1 = df.dropna(inplace=False)
df1
```

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

```
df1 = df.dropna(inplace=True)
df1
```

Creazione e Visualizzazione di DataFrame con Dati Mancanti

```
# Generazione di dati di esempio con valori mancanti
data = {
    'Variable1': [1, 2, 3, 4, 5],
    'Variable2': [1, 2, np.nan, 4, np.nan],
    'Missing_Column': ['A', 'B', 'A', 'C', np.nan]
}

# Creazione di un DataFrame dai dati generati
df = pd.DataFrame(data)

# Inizializzazione di un nuovo DataFrame vuoto
df1 = pd.DataFrame()

# Visualizzazione del DataFrame creato
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
```

Selezione e Visualizzazione Colonne Numeriche

```
# Selezione delle colonne numeriche dal DataFrame
numeric_cols = df.select_dtypes(include=['number'])

# Visualizzazione dei nomi delle colonne numeriche
numeric_cols.columns
```

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

Riempi Valori Mancanti con Media

```
# Riempi i valori mancanti nelle colonne numeriche con la media delle rispettive colonne
df1[numeric_cols.columns] =
df[numeric_cols.columns].fillna(df[numeric_cols.columns].mean())

# Visualizzazione del DataFrame aggiornato
df1
```

```
   Variable1  Variable2
0          1   1.000000
1          2   2.000000
```

```
2           3   2.333333
3           4   4.000000
4           5   2.333333
```

Selezione e Visualizzazione Colonne Categoriche

```
# Selezione delle colonne categoriche dal DataFrame
categorical_cols = df.select_dtypes(exclude=['number'])

# Visualizzazione dei nomi delle colonne categoriche
categorical_cols.columns
```

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

Rimpiazzo Valori Mancanti con Moda

```
# Riempi i valori mancanti nelle colonne categoriche con la moda delle
rispettive colonne
df1[categorical_cols.columns] =
df[categorical_cols.columns].fillna(df[categorical_cols.columns].mode(
).iloc[0])

# Visualizzazione del DataFrame aggiornato
df1
```

```
   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
```

Generazione DataFrame con Valori Mancanti

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

# Generazione di un DataFrame con dati di esempio, inclusi valori
mancanti
data = {
    'Feature1': [1, 2, np.nan, 4, 5],  # np.nan indica valori mancanti
    'Feature2': [np.nan, 2, 3, 4, np.nan],
    'Feature3': [1, np.nan, 3, 4, 5]
}

# Creazione del DataFrame da un dizionario di dati
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
```

valori mancanti nel DF

```
# valori mancanti nel 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
```

Calcolo Percentuale Valori Mancanti

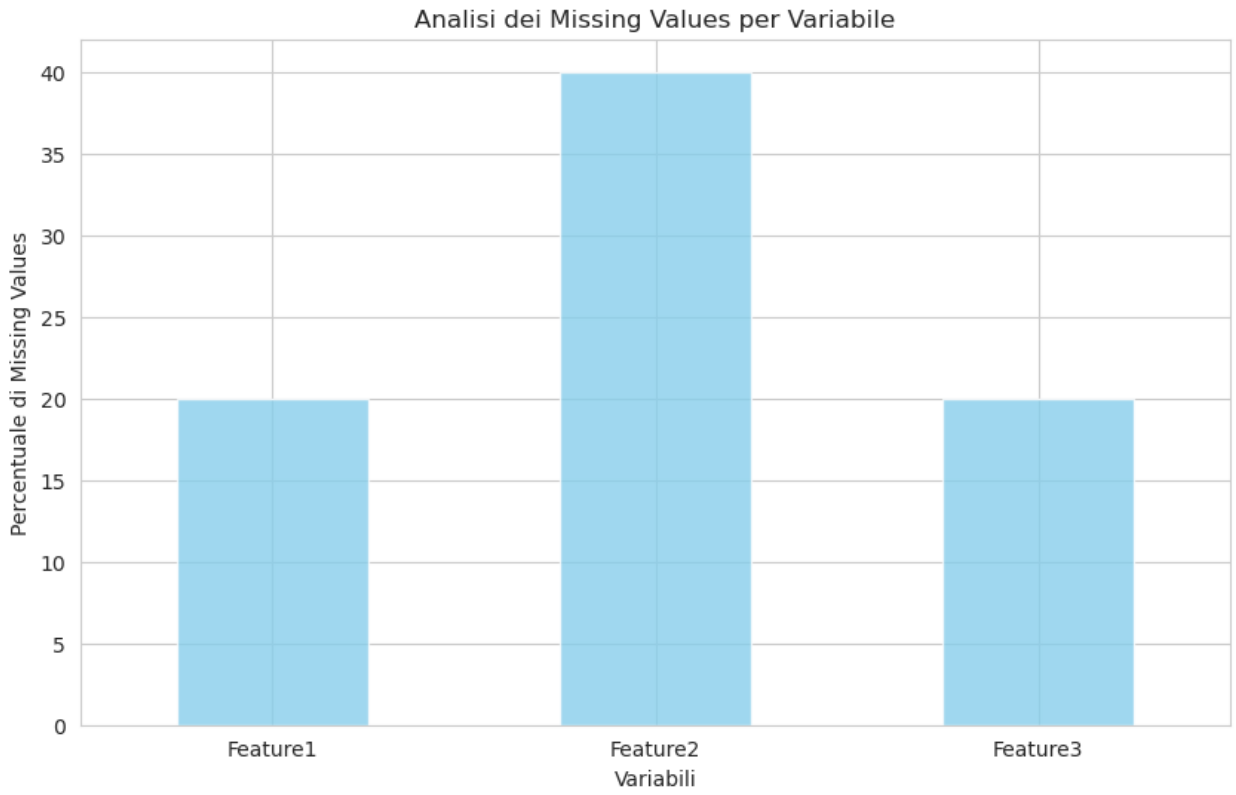
```
# Calcolo della percentuale di valori mancanti per ogni colonna nel DataFrame
missing_percent = (df.isnull().sum() / len(df)) * 100
missing_percent
```

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

I Grafici Per I Missing Values

Visualizzazione Percentuale Missing Values

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

# Creazione del grafico a barre per visualizzare la percentuale di valori mancanti per variabile
plt.figure(figsize=(10, 6))
missing_percent.plot(kind='bar', color='skyblue', alpha=0.8)
plt.xlabel('Variabili')
plt.ylabel('Percentuale di Missing Values')
plt.title('Analisi dei Missing Values per Variabile')
plt.xticks(rotation=0)
plt.show()
```



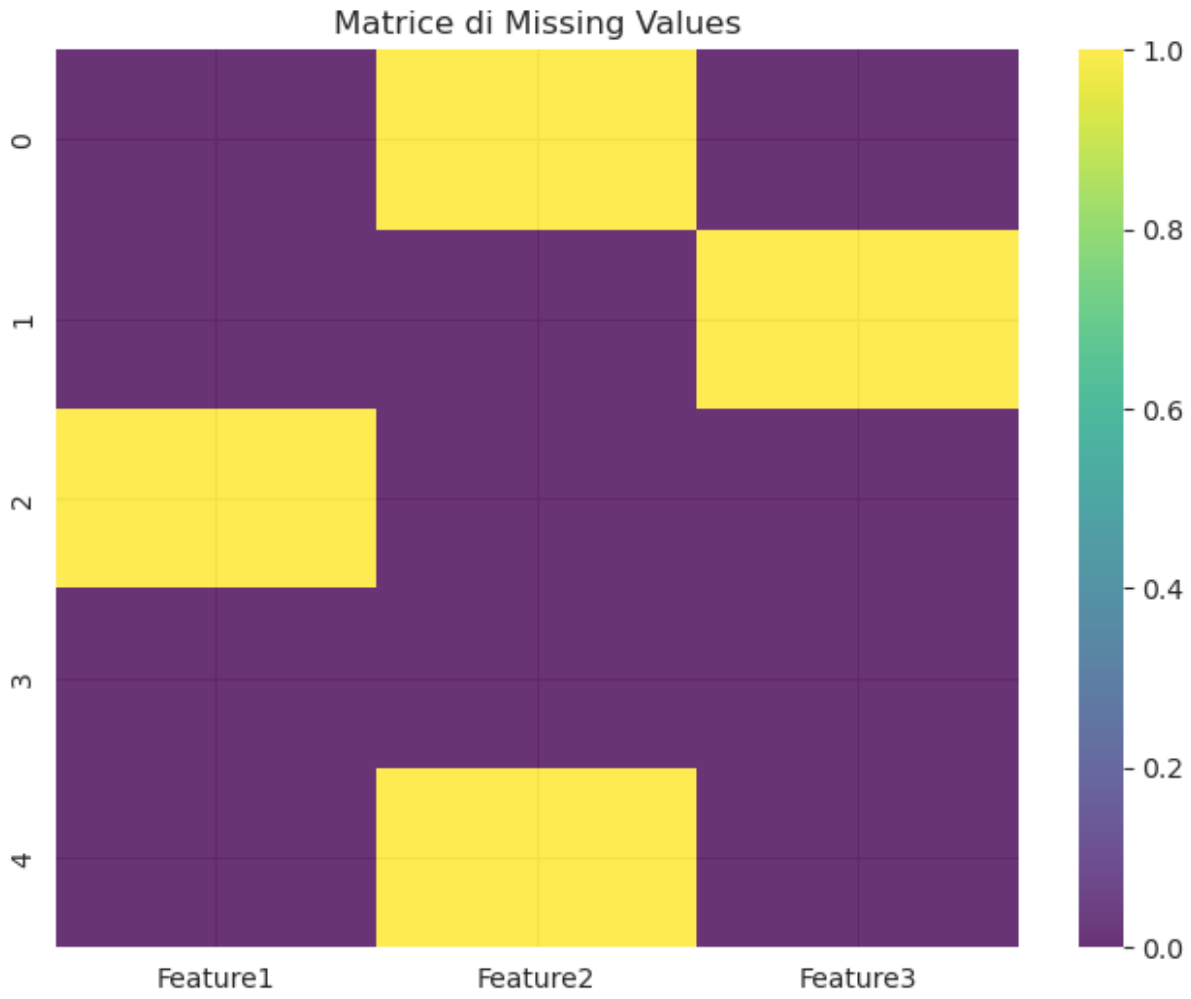
Generazione e Analisi di Missing Values

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

# Creazione di un DataFrame dai dati generati
df = pd.DataFrame(data)

# Calcolo della matrice di valori mancanti nel DataFrame
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

Visualizzazione Matrice Missing Values con Heatmap

```
# Creazione di una heatmap per visualizzare la matrice di missing values
plt.figure(figsize=(8, 6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=True, alpha=0.8)
plt.title('Matrice di Missing Values')
plt.show()
```



Generazione e Anteprima Dati Casuali

```
import numpy as np
import pandas as pd

# Generazione di dati casuali per l'esplorazione
np.random.seed(42)  # Imposta il seed per la riproducibilità
data = {
    'Età': np.random.randint(18, 70, size=1000),  # Età random tra 18 e 70
```

```
    'Genere': np.random.choice(['Maschio', 'Femmina'], size=1000),  # Genere casuale
    'Punteggio': np.random.uniform(0, 100, size=1000),  # Punteggio casuale tra 0 e 100
    'Reddito': np.random.normal(50000, 15000, size=1000)  # Reddito con distribuzione normale
}

df = pd.DataFrame(data)

# Visualizza le prime righe del dataset per un'anteprima
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
```

Analisi Esplorativa Iniziale del DataFrame

```
# Visualizzazione delle informazioni generali sul DataFrame
print(df.info())

# Visualizzazione delle statistiche descrittive del DataFrame
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
```

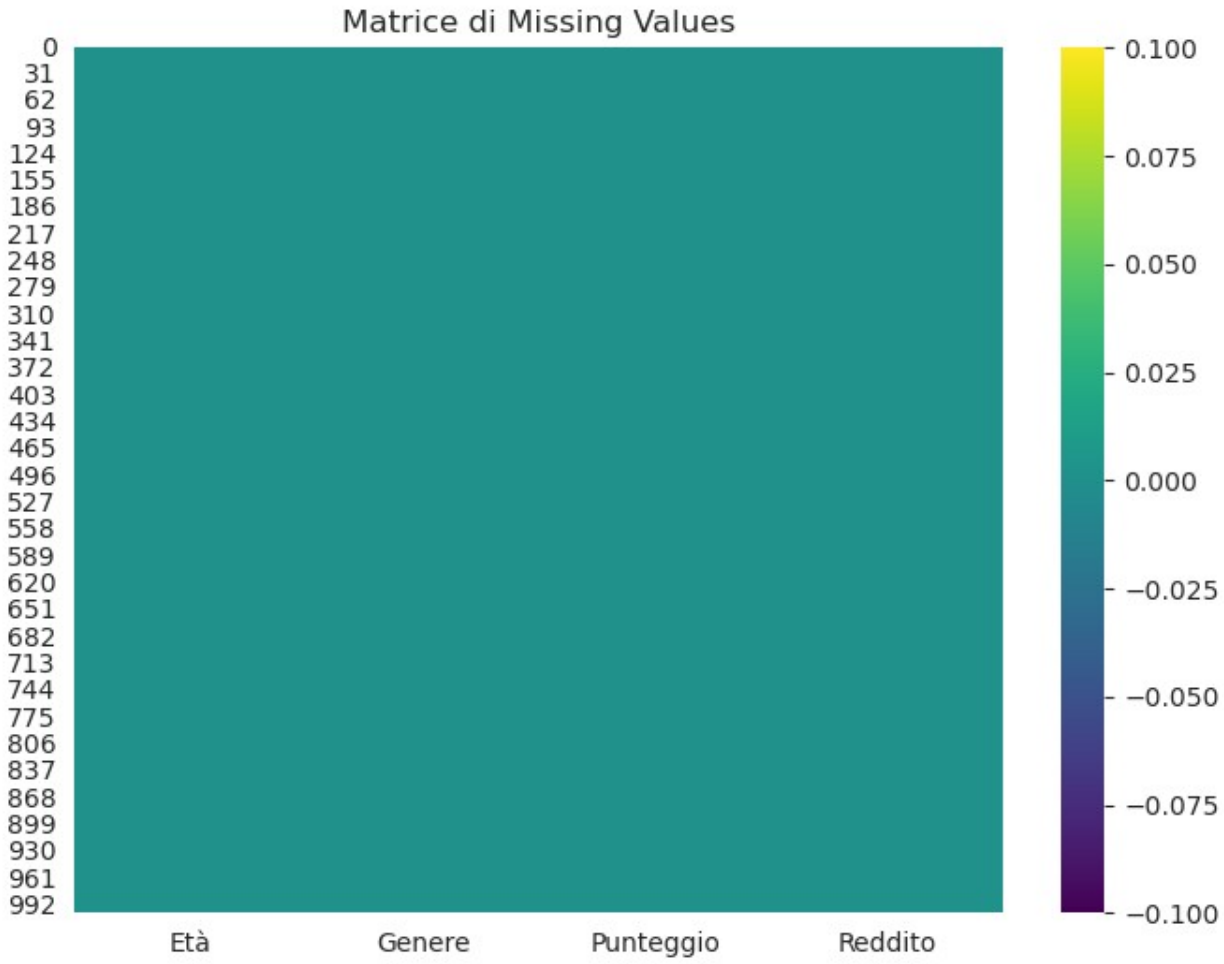
Riepilogo Valori Mancanti

```
# Calcolo dei valori mancanti per ogni colonna nel DataFrame
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
```

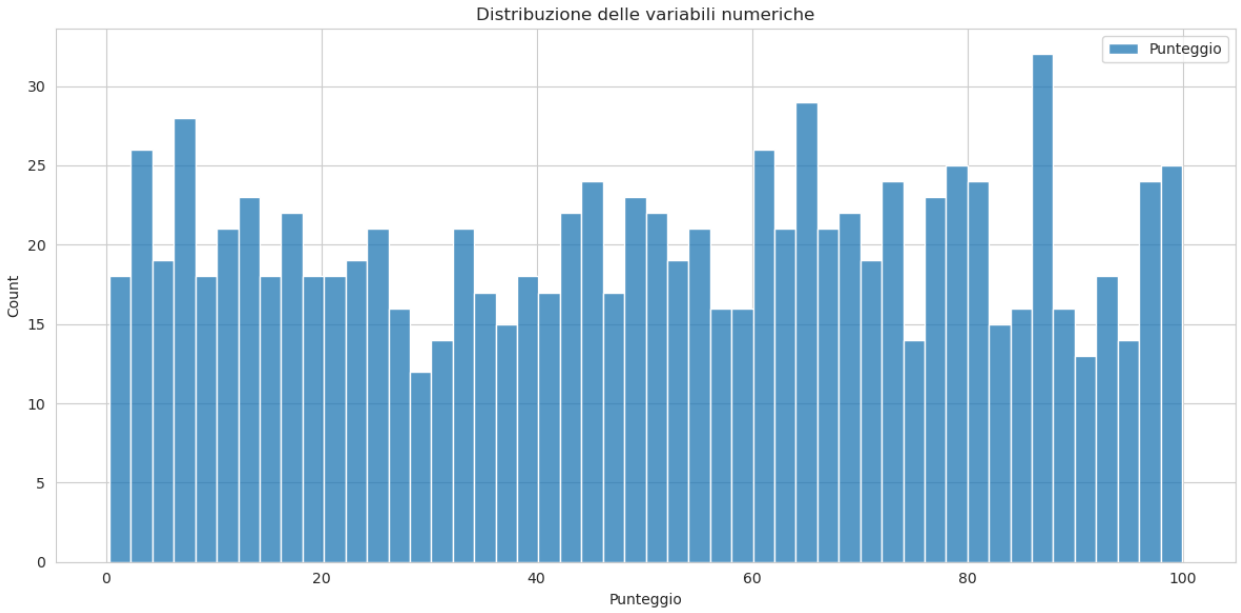
Heatmap di Missing Values senza Barra Colori

```
# Visualizzazione di una heatmap per i valori mancanti nel DataFrame
plt.figure(figsize=(8, 6))
sns.heatmap(df.isnull(), cmap='viridis', cbar=True)
plt.title('Matrice di Missing Values')
plt.show()
```



Visualizzazione Distribuzione Variabili Numeriche

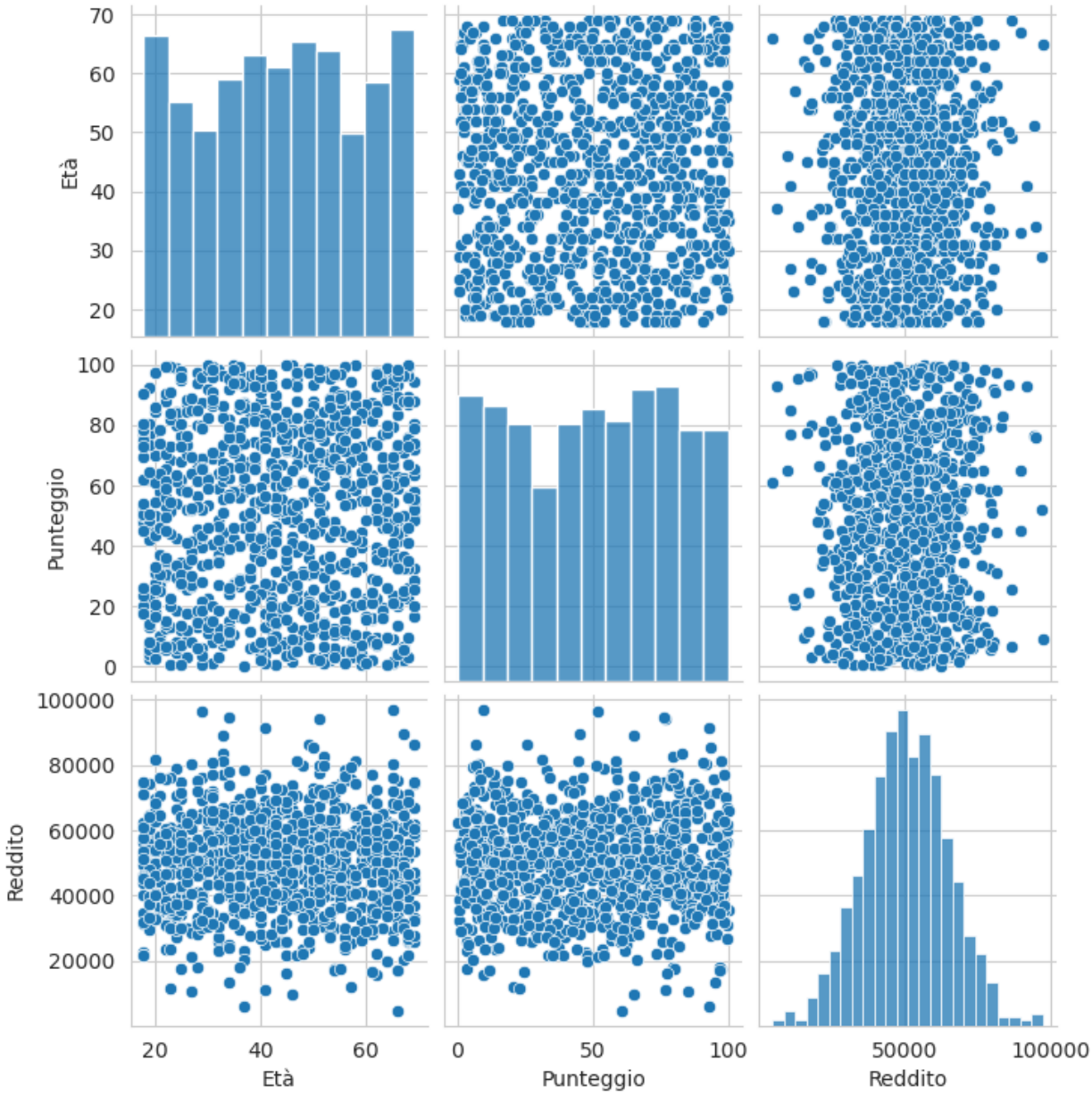
```
# Visualizzazione della distribuzione della variabile "Punteggio"
plt.figure(figsize=(12, 6))
sns.set_style("whitegrid")
sns.histplot(df["Punteggio"], kde=False, bins=50, label="Punteggio")
plt.legend()
plt.title('Distribuzione delle variabili numeriche')
plt.tight_layout()  # Ottimizza il layout del grafico
plt.show()
```



Esplorazione Grafica delle Variabili Numeriche

```
# Selezione delle caratteristiche numeriche del DataFrame
numeric_features = df.select_dtypes(include=[np.number])

# Creazione di una matrice di scatter plot per le variabili numeriche
sns.pairplot(df[numeric_features.columns])

# Visualizzazione dei grafici
plt.show()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```



Confronto Punteggio per Genere con Box Plot

```
# Creazione di un box plot per confrontare i punteggi tra i diversi generi
plt.figure(figsize=(10, 6))
sns.boxplot(x='Genere', y='Punteggio', data=df)
plt.title('Box Plot tra Genere e Punteggio')
plt.show()
```

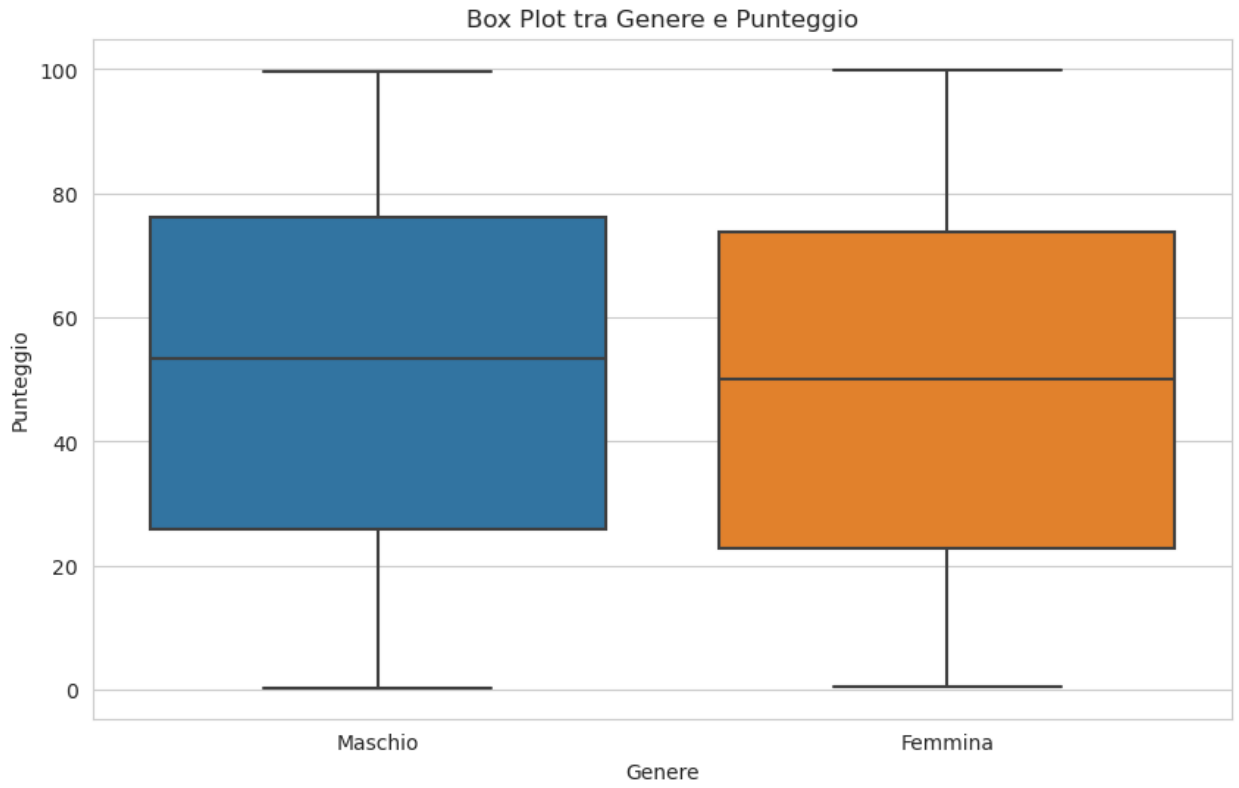
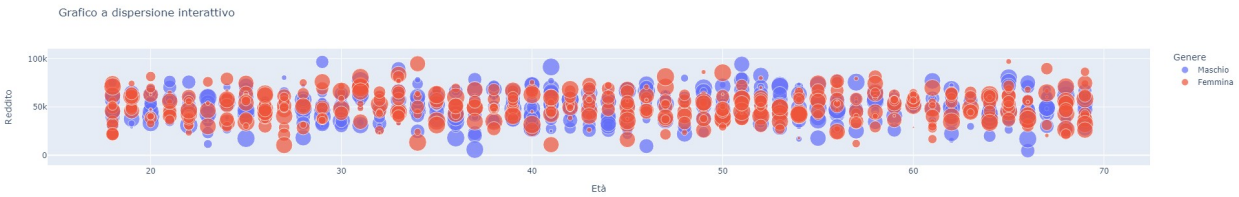


Grafico a Dispersione Interattivo

```
# Creazione di un grafico a dispersione interattivo con Plotly Express
fig = px.scatter(df, x='Età', y='Reddito', color='Genere',
size='Punteggio',
                 title='Grafico a dispersione interattivo')

# Aggiornamento del layout del grafico
fig.update_layout(title='Grafico a dispersione interattivo')

# Visualizzazione del grafico
fig.show()
```



Generazione e Anteprima di Dati Casuali

```
# Inizializzazione del seed per la riproducibilità dei dati casuali
np.random.seed(42)
```

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

# Creazione di un DataFrame con i dati generati
df = pd.DataFrame(data)

# Visualizzazione delle prime righe del dataset per un'anteprima
print(df.head())
```

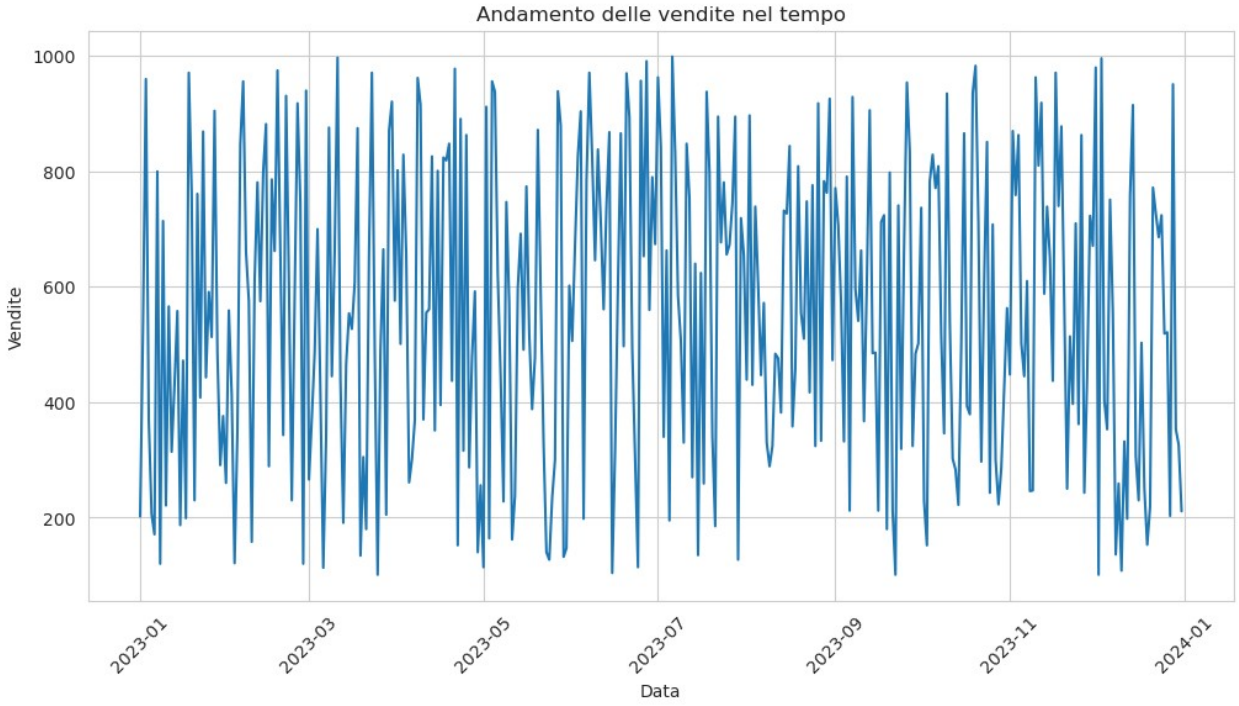
```
        Data  Vendite Prodotto
0 2023-01-01      202        B
1 2023-01-02      535        A
2 2023-01-03      960        C
3 2023-01-04      370        A
4 2023-01-05      206        A
```

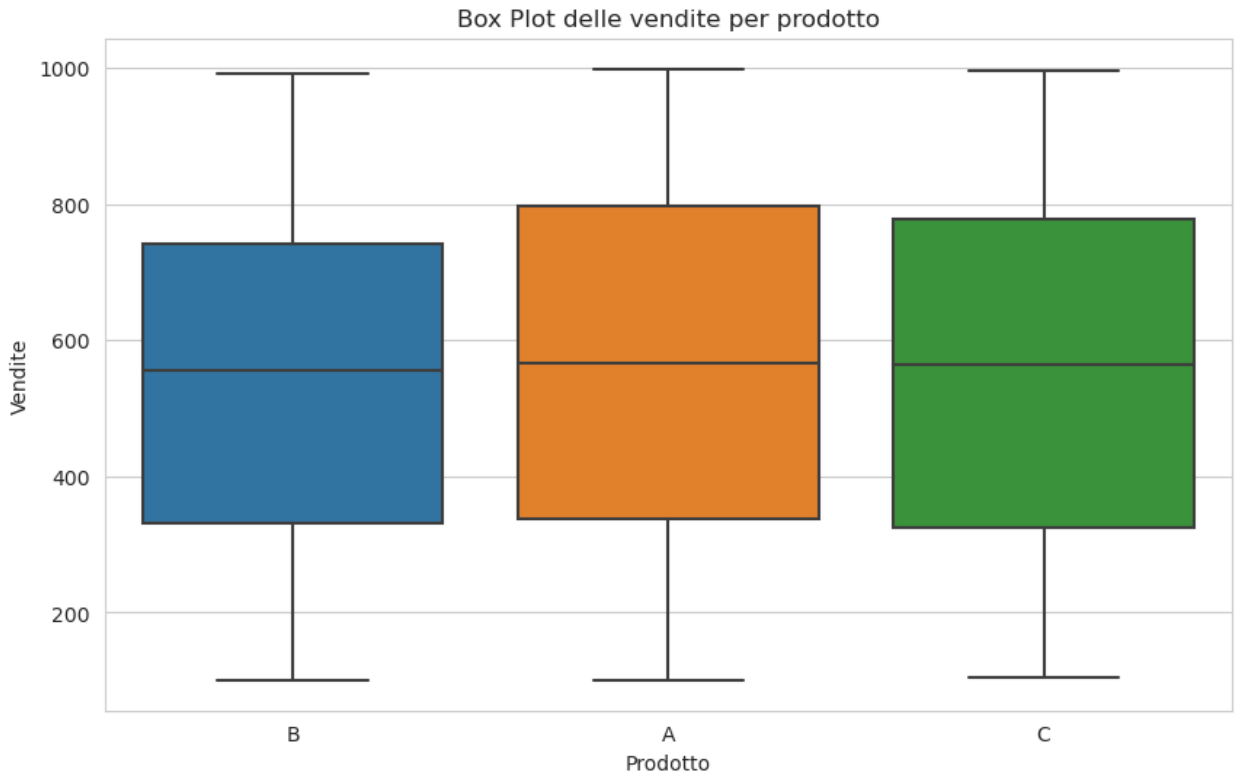
Analisi Esplorativa dei Dati

```
# Impostazione del seed per generare dati casuali riproducibili
np.random.seed(42)

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

# Creazione di un DataFrame dai dati generati
df = pd.DataFrame(data)

# Visualizzazione delle prime righe del dataset per verificare i dati
print(df.head())

# Visualizzazione di 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()

# Visualizzazione di un 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()
```

	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





Analisi Esplorativa dei Dati di Vendita

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

# Creazione di un DataFrame con i dati generati
df = pd.DataFrame(data)

# Visualizzazione del DataFrame
print(df)
```

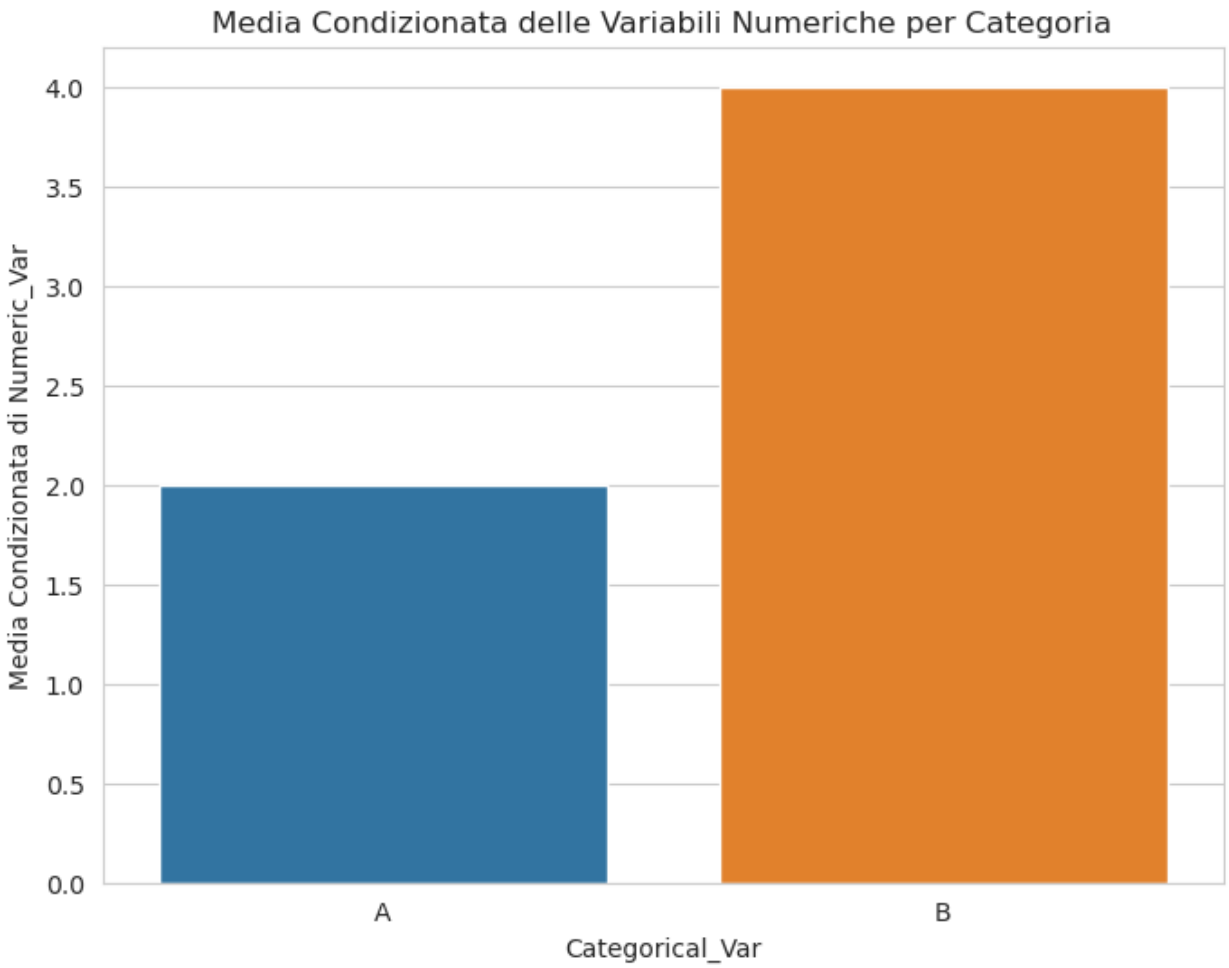
```
   Numeric_Var Categorical_Var
0          1.0              A
1          2.0              B
2          3.0              A
3          4.0              B
4          NaN              A
5          6.0              B
```

Calcolo e Visualizzazione della Media Condizionata

```
# Calcolo della media condizionata per 'Numeric_Var' basata su
'Categorical_Var'
```

```
conditional_means =
df['Numeric_Var'].fillna(df.groupby('Categorical_Var')
['Numeric_Var'].transform('mean'))

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

# Creazione di un grafico a barre per mostrare la media condizionata
per ogni categoria
plt.figure(figsize=(8,6))
sns.barplot(data=df, x='Categorical_Var', y='Numeric_Var',
errorbar=None)
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
```

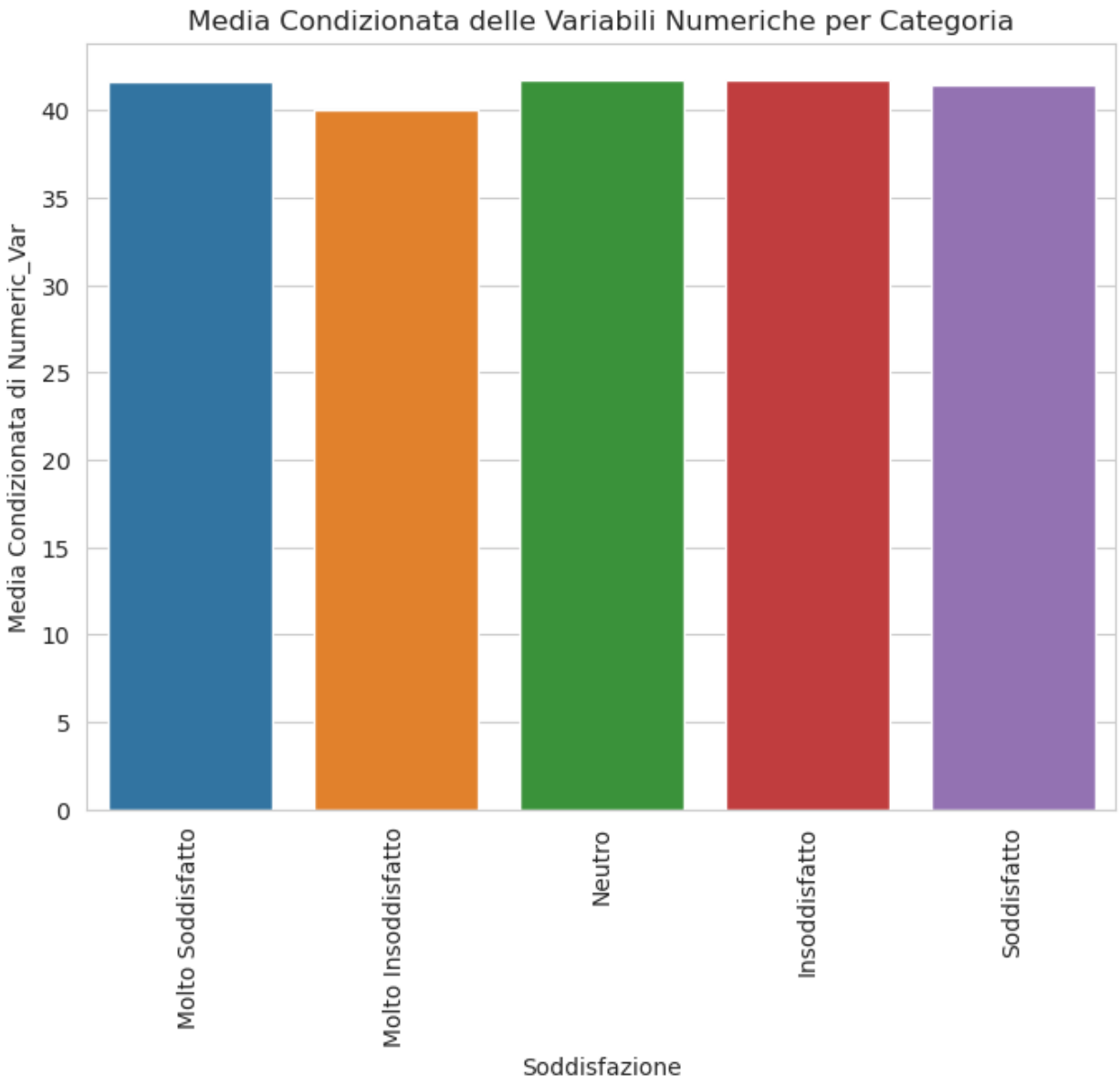


Calcolo e Visualizzazione della Media Condizionata

```
# Genera dati casuali per l'esplorazione
#np sta per numeri pytone
#se non metto il dato seed mi danno sempre i numeri casuali:D:
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]
```

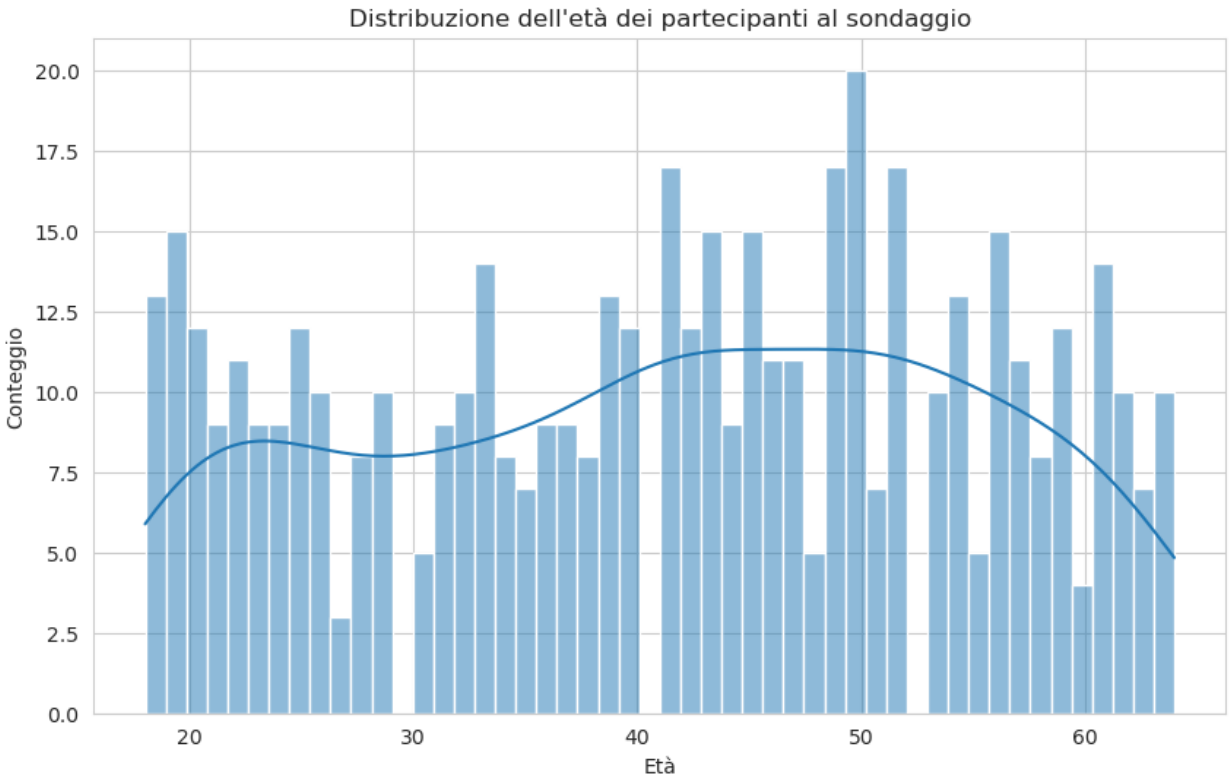


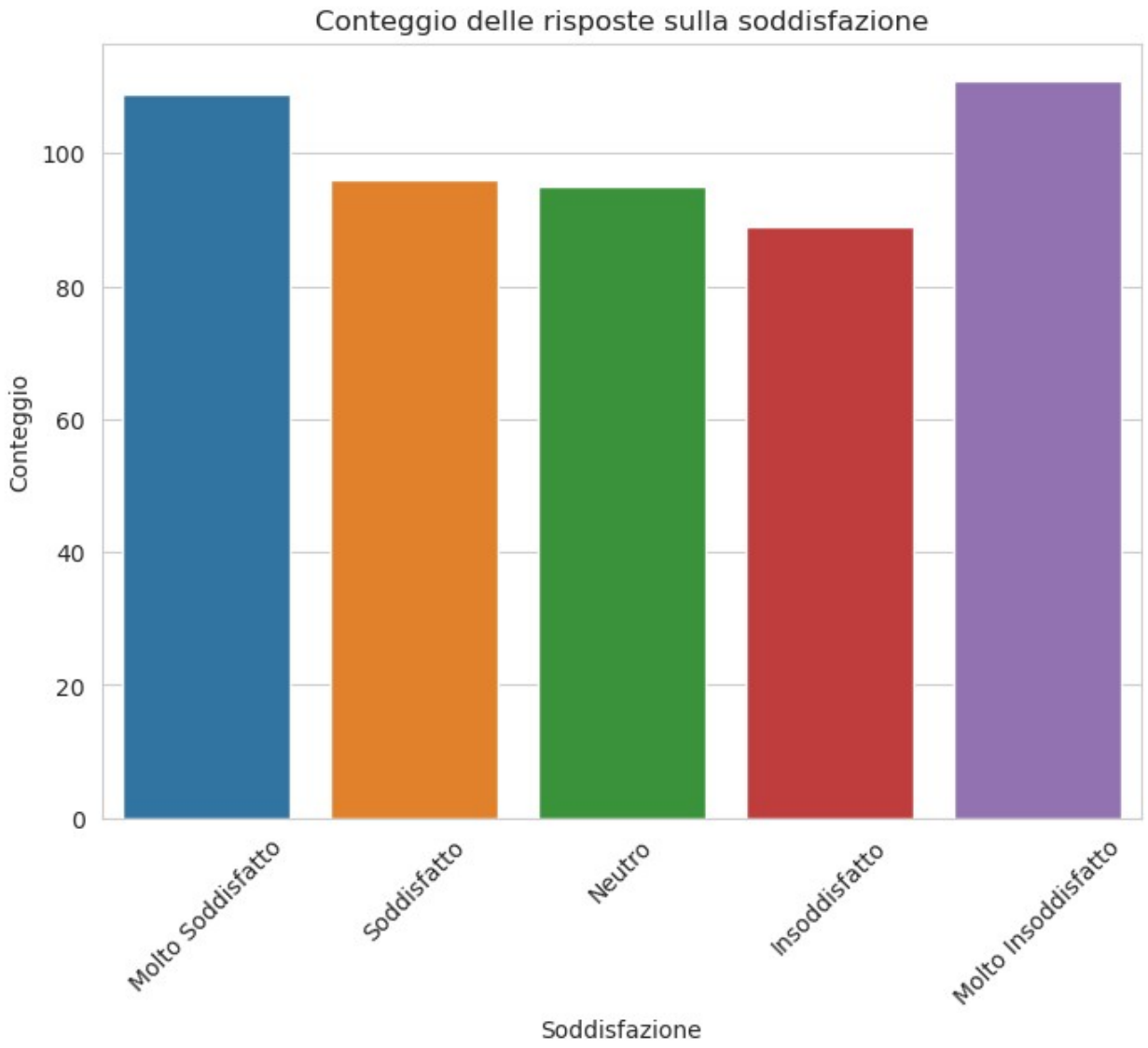
Calcolo e Visualizzazione dell'età e della soddisfazione

```
# Visualizzazione delle prime righe del DataFrame per un'anteprima dei dati
print(df.head())

# Creazione di un grafico per la distribuzione dell'età
plt.figure(figsize=(10, 6))
sns.histplot(df['Età'], bins=50, kde=True)  # KDE per una stima della densità
plt.title('Distribuzione dell\'età dei partecipanti al sondaggio')
plt.xlabel('Età')
plt.ylabel('Conteggio')
plt.show()
```

```
# Creazione di un grafico a barre per le 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





Generazione e Introduzione di Valori Mancanti

```
# Impostazione del seed per garantire la riproducibilità dei risultati
np.random.seed(37)

# Creazione di un DataFrame vuoto
df = pd.DataFrame()

# Generazione di dati casuali per il DataFrame
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)
```

```
# Calcolo del numero totale di valori mancanti desiderati (3% del totale)
total_missing_values = int(0.03 * n_rows * len(df.columns))

# Introduzione casuale di valori mancanti nel DataFrame
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       X  0.505056     33.0  0.295902
1          A       X -0.565473     59.0  0.020742
2          B       Y -0.302934      3.0  0.010785
3          C       Y -0.893455     42.0  0.029415
4          C       Y  0.419053     45.0  0.284421
...      ...     ...       ...      ...       ...
9995     NaN       Y       NaN     98.0  0.760199
9996       A       Y  0.340934      NaN  0.724716
9997       A       X  0.022344     53.0  0.667575
9998       C       X -0.247306     20.0  0.146930
9999       B       Y  1.985091     71.0  0.033294

[10000 rows x 5 columns]
```

Calcolo Righe con Dati Mancanti

```
# Calcolo del numero di righe con almeno un valore mancante
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
len(righe_con_dati_mancanti)
```

```
3648
```

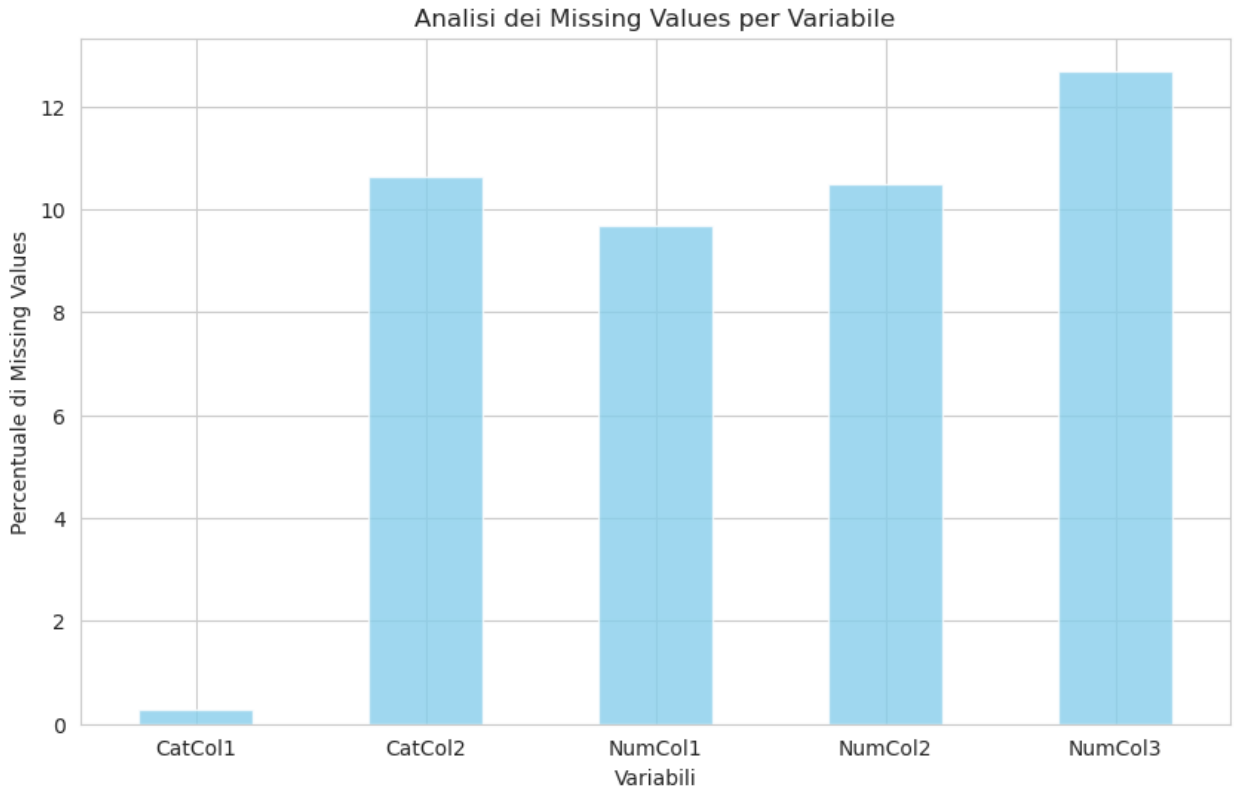
Calcolo Percentuale dei Missing Values in Ogni Colonna del DataFrame

```
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
```

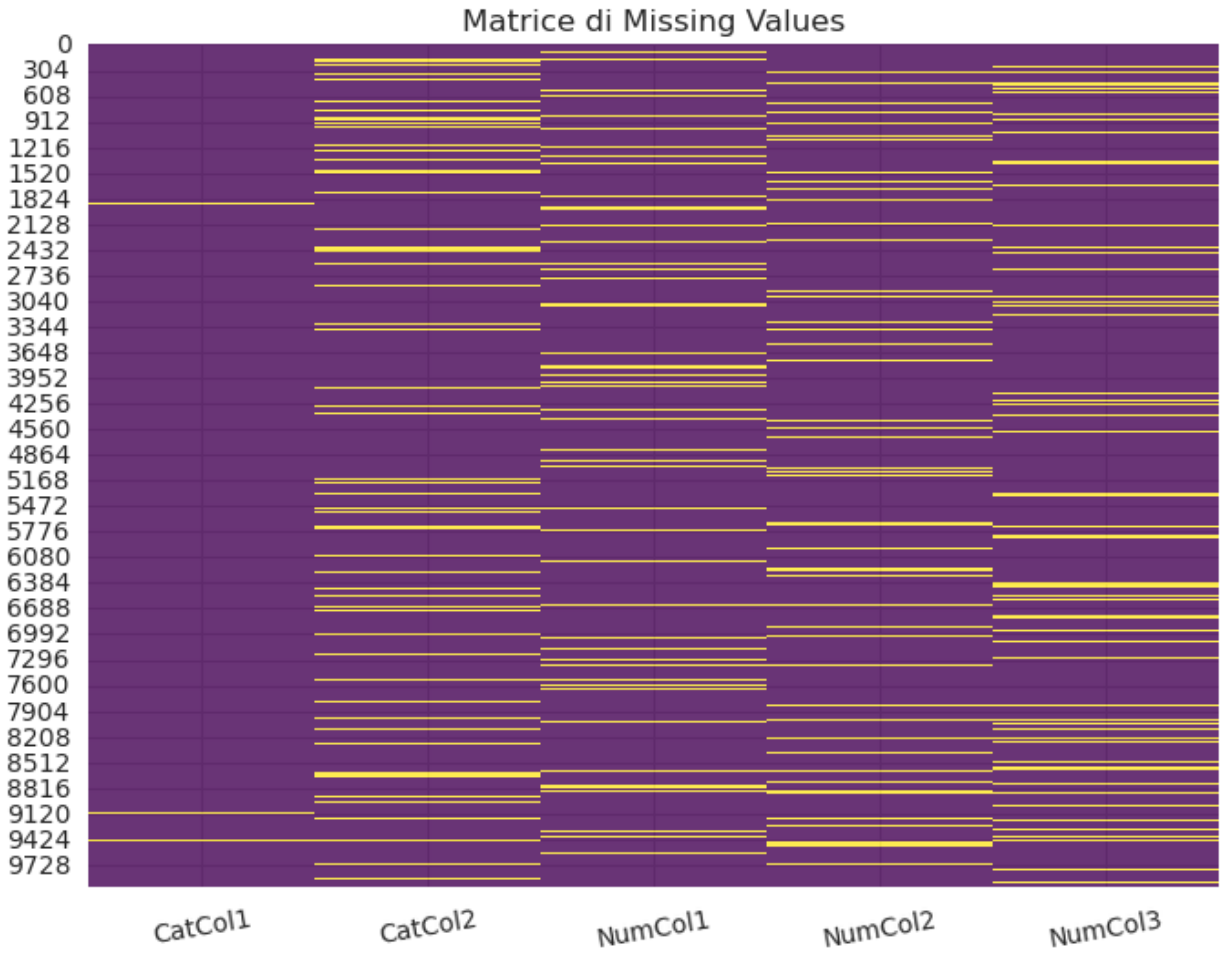
Visualizzazione dei Missing Values per Variabile

```
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('Variabili')
plt.ylabel('Percentuale di Missing Values')
plt.title('Analisi dei Missing Values per Variabile')
plt.xticks(rotation=0)
plt.show()
```



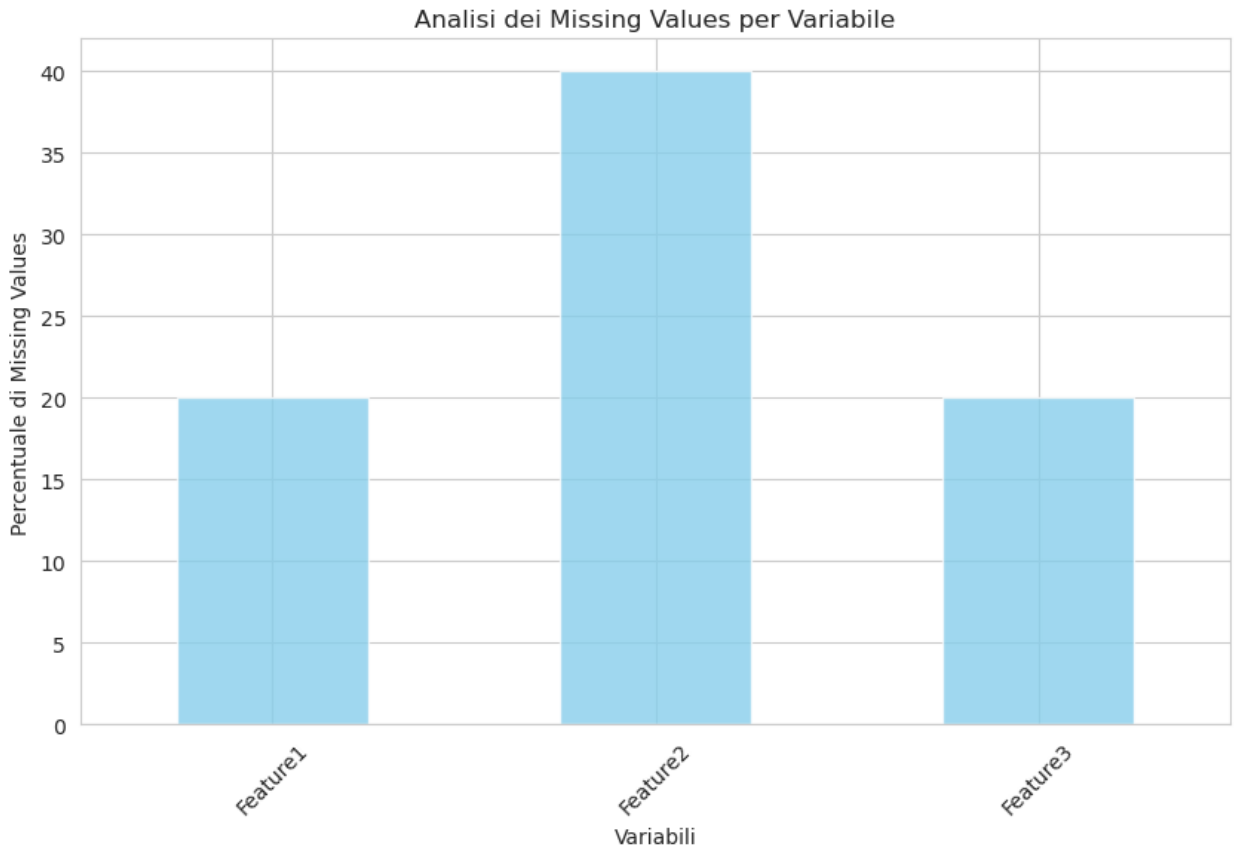
Visualizzazione Matrice Missing Values con Heatmap

```
#generazione della heatmap
missing_matrix = df.isnull()

plt.figure(figsize=(8, 6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
plt.title('Matrice di Missing Values')
plt.xticks(rotation=10)

plt.show()
```



Visualizzazione Percentuale Missing Values

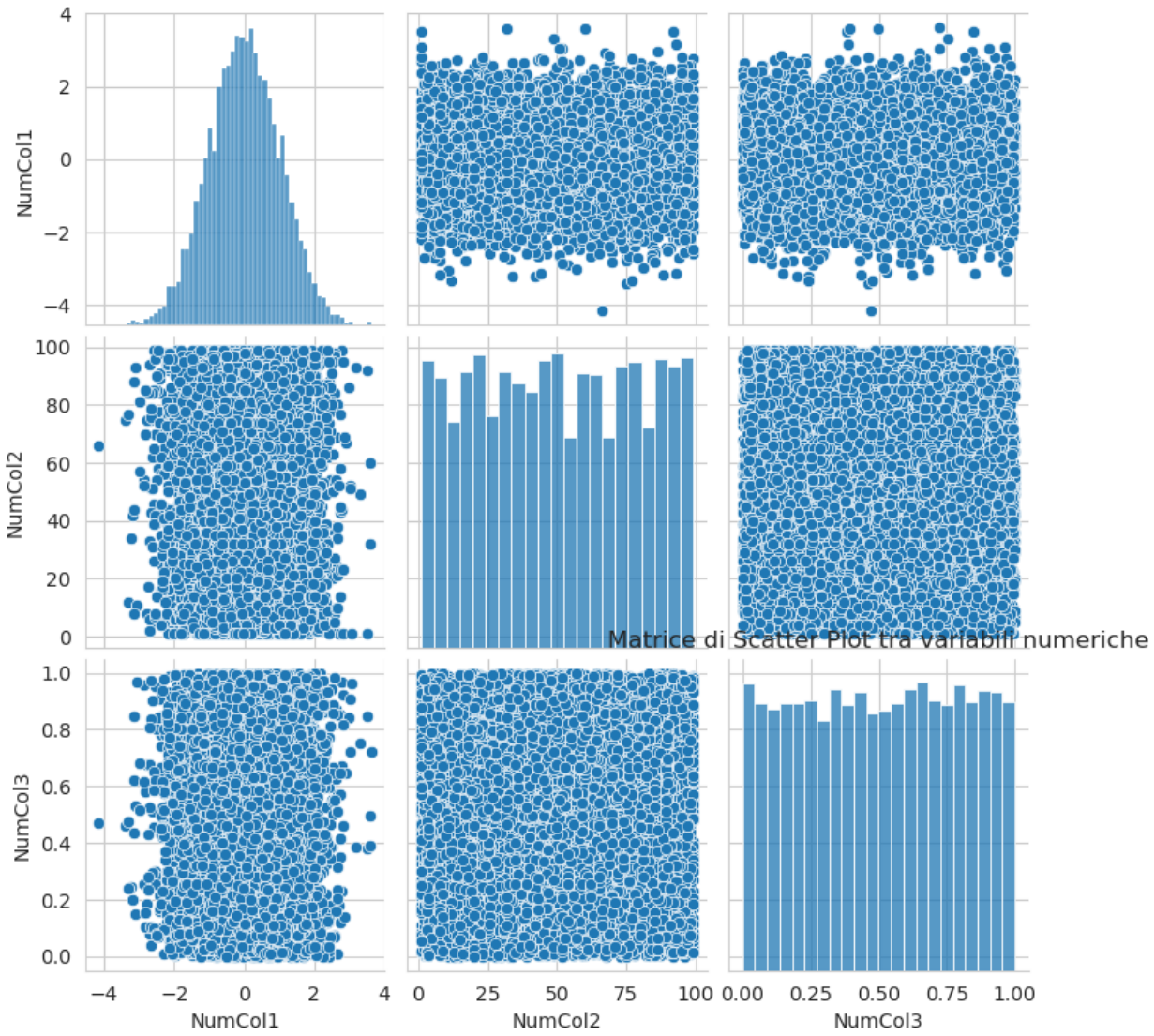
```
# Crea il grafico a barre per visualizzare la percentuale di valori mancanti per variabile
plt.figure(figsize=(10, 6))
missing_percent.plot(kind='bar', color='skyblue', alpha=0.8)
plt.xlabel('Variabili')
plt.ylabel('Percentuale di Missing Values')
plt.title('Analisi dei Missing Values per Variabile')
plt.xticks(rotation=45)
plt.show()
```



Visualizzazione Matrice di Scatter Plot

```
# Selezione delle colonne numeriche dal DataFrame
numeric_features = df.select_dtypes(include=[np.number])

# Creazione della matrice di scatter plot tra variabili numeriche
sns.pairplot(df[numeric_features.columns])

# Aggiunta del titolo al grafico
plt.title('Matrice di Scatter Plot tra variabili numeriche')

# Visualizzazione del grafico
plt.show()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```



Rimozione Righe con Valori Mancanti in Colonne Categoriche Specifiche

```
# Rimuovi le righe con valori mancanti nelle colonne specificate
df = df.dropna(subset=["CatCol1", "CatCol2"], how="all")
df
```

```
     CatCol1 CatCol2   NumCol1  NumCol2   NumCol3
0          A       X  0.505056     33.0  0.295902
1          A       X -0.565473     59.0  0.020742
2          B       Y -0.302934      3.0  0.010785
3          C       Y -0.893455     42.0  0.029415
4          C       Y  0.419053     45.0  0.284421
...      ...     ...       ...      ...       ...
9995     NaN       Y       NaN     98.0  0.760199
```

```
9996       A       Y  0.340934      NaN  0.724716
9997       A       X  0.022344     53.0  0.667575
9998       C       X -0.247306     20.0  0.146930
9999       B       Y  1.985091     71.0  0.033294

[9907 rows x 5 columns]
```

Rimozione Righe con Valori Mancanti in Colonne Numeriche Specifiche

```
# Rimuovi le righe con valori mancanti nelle colonne specificate
df = df.dropna(subset=["NumCol1", "NumCol2", "NumCol3"], how="all")
df
```

```
     CatCol1 CatCol2   NumCol1  NumCol2   NumCol3
0          A       X  0.505056     33.0  0.295902
1          A       X -0.565473     59.0  0.020742
2          B       Y -0.302934      3.0  0.010785
3          C       Y -0.893455     42.0  0.029415
4          C       Y  0.419053     45.0  0.284421
...      ...     ...       ...      ...       ...
9995     NaN       Y       NaN     98.0  0.760199
9996       A       Y  0.340934      NaN  0.724716
9997       A       X  0.022344     53.0  0.667575
9998       C       X -0.247306     20.0  0.146930
9999       B       Y  1.985091     71.0  0.033294

[9899 rows x 5 columns]
```

Pulizia e Imputazione Dati Mancanti

```
# Selezione delle colonne numeriche e categoriche dal DataFrame
numeric_cols = df.select_dtypes(include=['number'])
categorical_cols = df.select_dtypes(exclude=['number'])

# Riempimento dei valori mancanti nelle colonne categoriche con la
moda
df.loc[:, categorical_cols.columns] =
df[categorical_cols.columns].fillna(df[categorical_cols.columns].mode(
).iloc[0])

# Calcolo della media condizionale per le colonne numeriche,
raggruppando per 'CatCol1'
conditional_means =
df[numeric_cols.columns].fillna(df.groupby('CatCol1')
[numeric_cols.columns].transform('mean'))

# Sostituzione dei valori mancanti nelle colonne numeriche con le
medie condizionali calcolate
```

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

# Visualizzazione del DataFrame aggiornato
print(df)
```

```
     CatCol1 CatCol2   NumCol1    NumCol2   NumCol3
0          A       X  0.505056  33.000000  0.295902
1          A       X -0.565473  59.000000  0.020742
2          B       Y -0.302934   3.000000  0.010785
3          C       Y -0.893455  42.000000  0.029415
4          C       Y  0.419053  45.000000  0.284421
...      ...     ...       ...        ...       ...
9995       A       Y  0.003936  98.000000  0.760199
9996       A       Y  0.340934  50.080695  0.724716
9997       A       X  0.022344  53.000000  0.667575
9998       C       X -0.247306  20.000000  0.146930
9999       B       Y  1.985091  71.000000  0.033294

[9899 rows x 5 columns]
```

FINE DELLA PARTE DEI MISSING VALUES

Lavoro fatto da:

Alessandro

Kozar

Rossi