MISSING VALUES

KEVIN;DE SANTIS; BONANNO;DEL ORTO

Generazione e Visualizzazione di Dati Casuali con Pandas e NumPy

```
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),
     #random normal esce il numero con la media più alta, invece
quella più bassa ha poca possibilità di uscire
    'Reddito': np.random.normal(50000, 15000, size=1000)
}

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
```

il codice crea un DataFrame con colonne rappresentanti età, genere, punteggio e reddito, tutte con dati casuali. Questo può essere utile per scopi di esplorazione dati o per testare visualizzazioni e analisi.

Creazione e Anteprima di un DataFrame Pandas con Dati Casuali

```
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
```

```
    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
```

Il codice crea un DataFrame utilizzando un elenco di dizionari. Ogni dizionario rappresenta una riga del DataFrame con le seguenti colonne:

Età: Rappresenta l'età della persona. Punteggio: Indica un punteggio associato alla persona. Ammesso: È un valore binario (1 o 0) che indica se la persona è stata ammessa o meno. In breve, il codice crea un DataFrame con dati mancanti rappresentati da None o NaN. Questo può essere utile per esplorare come gestire i dati mancanti in analisi o modelli statistici.

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
```

Il codice accede alla colonna "punteggio" nel DataFrame "df". Questo significa che stai estraendo una Serie di Pandas contenente i valori presenti nella colonna "punteggio".

Per spiegare in dettaglio:

"df" è il DataFrame creato in precedenza. Le parentesi quadre [] vengono utilizzate per accedere a una colonna specifica all'interno del DataFrame. All'interno delle parentesi quadre, "punteggio" è il nome della colonna a cui vogliamo accedere. Il risultato di questo codice è una Serie di Pandas contenente i valori dalla colonna "punteggio". Puoi eseguire diverse operazioni su questa Serie, come calcolare statistiche, creare grafici o filtrare i dati.

La matrice di valori mancanti

```
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

Il codice crea un DataFrame utilizzando un elenco di dizionari. Ogni dizionario rappresenta una riga del DataFrame con le seguenti colonne:

Feature1: Contiene i valori 1, 2, NaN, 4 e 5. Feature2: Contiene i valori NaN, 2, 3, 4 e NaN. Feature3: Contiene i valori 1, NaN, 3, 4 e 5. Successivamente, viene calcolata la matrice di valori mancanti (missing values) per il DataFrame. La matrice risultante ha valori booleani (True o False), dove True indica che il valore è mancante (NaN) e False indica che il valore è presente.

La matrice di valori mancanti

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

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

Il codice crea un nuovo DataFrame chiamato righe_con_dati_mancanti. Questo DataFrame contiene solo le righe del DataFrame originale (df) che contengono almeno un valore mancante

(NaN) in una qualsiasi delle colonne. In breve, il codice seleziona le righe che hanno almeno un valore mancante e le memorizza nel DataFrame righe_con_dati_mancanti.

Calcolo del totale dei dati mancanti nelle righe

```
totale_dati_mancanti = righe_con_dati_mancanti.shape[1]
totale_dati_mancanti
```

```
3
```

Il codice calcola il totale dei dati mancanti nelle righe selezionate del DataFrame. Ecco una spiegazione dettagliata:

righe_con_dati_mancanti: Questo è un DataFrame che contiene solo le righe del DataFrame originale (df) che contengono almeno un valore mancante (NaN) in una qualsiasi delle colonne. shape[1]: La proprietà shape restituisce una tupla con le dimensioni del DataFrame. In particolare, shape[1] restituisce il numero di colonne nel DataFrame righe_con_dati_mancanti. Quindi, totale_dati_mancanti conterrà il numero totale di valori mancanti nelle colonne delle righe selezionate

Visualizzazione delle Righe con Dati Mancanti e Calcolo del Totale

```
print("righe con dati mancanti:")
print(righe_con_dati_mancanti)
print("totale dati mancanti: ", totale_dati_mancanti)
```

```
righe con dati mancanti:
   Feature1  Feature2  Feature3
0       1.0       NaN       1.0
1       2.0       2.0       NaN
2       NaN       3.0       3.0
4       5.0       NaN       5.0
totale dati mancanti:  3
```

Il codice stampa le righe del DataFrame che contengono dati mancanti (NaN) e il totale dei dati mancanti.

print("righe con dati mancanti:"): Questa riga stampa un'intestazione per le righe con dati mancanti. print(righe_con_dati_mancanti): Questa riga stampa le righe del DataFrame righe_con_dati_mancanti che contengono valori mancanti. print("totale dati mancanti: ", totale_dati_mancanti): Questa riga stampa il conteggio totale dei dati mancanti nel DataFrame.

Creazione di un DataFrame da un Dataset con Dati Mancanti

```
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
```

Il codice crea un DataFrame utilizzando un elenco di dizionari. Ogni dizionario rappresenta una riga del DataFrame con le seguenti colonne:

nome: Contiene i nomi delle persone. età: Rappresenta l'età della persona. punteggio: Indica un punteggio associato alla persona. email: Contiene gli indirizzi email delle persone. In breve, il codice crea un DataFrame con dati mancanti rappresentati da None o NaN. Questo può essere utile per esplorare come gestire i dati mancanti in analisi o modelli statistici.

Creazione di un Nuovo DataFrame Senza Righe Contenenti Dati Mancanti

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

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

Il codice esegue l'operazione di rimozione delle righe con dati mancanti (NaN) dal DataFrame df. Ecco una spiegazione dettagliata:

df.dropna(inplace=False): dropna() è un metodo di Pandas che rimuove le righe contenenti valori mancanti (NaN) da un DataFrame. L'argomento inplace=False indica che la modifica non viene effettuata direttamente sul DataFrame originale (df), ma viene restituito un nuovo DataFrame con le righe eliminate. Quindi, df1 è il nuovo DataFrame risultante dopo aver rimosso le righe con dati mancanti. df1: Questo DataFrame contiene tutte le righe del DataFrame originale df, tranne quelle con valori mancanti.

Visualizzazione della Matrice dei Dati Mancanti con Heatmap Colorata

```
# Crea una heatmap colorata
plt.figure(figsize=(8, 6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=True,alpha=0.8)
plt.title('Matrice di Missing Values')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

La heatmap colorata che hai creato rappresenta la matrice di missing values del DataFrame. Ecco una spiegazione dettagliata del codice:

plt.figure(figsize=(8, 6)): Questa riga crea una nuova figura (grafico) con una dimensione di 8x6 pollici. sns.heatmap(missing_matrix, cmap='viridis', cbar=True, alpha=0.8): sns.heatmap() è una funzione di Seaborn che crea una heatmap (mappa di calore) basata su una matrice di dati. L'argomento missing_matrix è la matrice di missing values creata in precedenza. cmap='viridis' specifica la mappa di colori da utilizzare (in questo caso, la scala di colori "viridis"). cbar=True

indica che vogliamo visualizzare una barra di colore sulla heatmap. alpha=0.8 regola la trasparenza della heatmap. plt.title('Matrice di Missing Values'): Questa riga imposta il titolo del grafico sulla heatmap. plt.show(): Questa riga mostra il grafico. La heatmap colorata evidenzia visivamente i valori mancanti (NaN) nel DataFrame. Le celle più chiare rappresentano i dati mancanti, mentre le celle più scure rappresentano i dati presenti.

Creazione di un DataFrame con Dati di Esempio

```
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

Il codice crea un DataFrame utilizzando un elenco di dizionari. Ogni dizionario rappresenta una riga del DataFrame con le seguenti colonne:

Variable1: Contiene i valori 1, 2, 3, 4 e 5. Variable2: Contiene i valori 1, 2, NaN, 4 e NaN. Missing_Column: Contiene i valori ‘A’, ‘B’, ‘A’, ‘C’ e NaN. In breve, il codice crea un DataFrame con dati mancanti rappresentati da NaN o None. Questo può essere utile per esplorare come gestire i dati mancanti in analisi o modelli statistici.

Selezione delle Colonnette Numeriche dal DataFrame

```
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

Il codice seleziona le colonne numeriche dal DataFrame df. Ecco una spiegazione dettagliata:

df.select_dtypes(include=['number']): Questo metodo seleziona le colonne del DataFrame df in base al tipo di dati. L'argomento include=['number'] indica che vogliamo selezionare solo le colonne con dati numerici (interi o float). Quindi, numeric_cols è un nuovo DataFrame contenente solo le colonne numeriche.

Colonnette Numeriche Selezionate dal DataFrame

```
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols.columns
```

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

Il codice seleziona le colonne numeriche dal DataFrame df. Ecco una spiegazione dettagliata:
df.select_dtypes(include=['number']): Questo metodo seleziona le colonne del DataFrame df in base al tipo di dati. L'argomento include=['number'] indica che vogliamo selezionare solo le colonne con dati numerici (interi o float). Quindi, numeric_cols è un nuovo DataFrame contenente solo le colonne numeriche.

Creazione di un DataFrame con Variabili Numeriche e Categoriche

```
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
```

Il codice crea un DataFrame utilizzando un elenco di dizionari. Ogni dizionario rappresenta una riga del DataFrame con le seguenti colonne:

Numeric_Var: Contiene i valori 1, 2, 3, 4, NaN e 6. Categorical_Var: Contiene i valori ‘A’, ‘B’, ‘A’, ‘B’, ‘A’ e ‘B’. In breve, il codice crea un DataFrame con dati mancanti rappresentati da NaN o None. Questo può essere utile per esplorare come gestire i dati mancanti in analisi o modelli statistici.

Creazione di un DataFrame con Dati Casuali e Valori Mancanti Simulati

```
import pandas as pd
import numpy as np


np.random.seed(41)


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)


total_missing_values = int(0.03 * n_rows * len(df.columns))


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]
```

Il codice crea un DataFrame con dati casuali e introduce valori mancanti (NaN) in alcune colonne. Ecco una spiegazione dettagliata:

Generazione dei dati casuali: Il DataFrame df viene inizializzato vuoto. Vengono generati dati casuali per le seguenti colonne: CatCol1: Contiene valori categorici (‘A’, ‘B’, ‘C’). CatCol2: Contiene valori categorici (‘X’, ‘Y’). NumCol1: Contiene valori numerici casuali. NumCol2: Contiene valori interi casuali. NumCol3: Contiene valori float casuali. Introduzione di valori mancanti: Viene calcolato il numero totale di valori mancanti da introdurre (3% del numero totale di righe moltiplicato per il numero di colonne). Per ogni colonna, vengono selezionati casualmente alcuni indici di riga per introdurre i valori mancanti (NaN). In breve, il codice crea un DataFrame con dati casuali e introduce valori mancanti in alcune colonne. Questo può essere utile per esplorare come gestire i dati mancanti in analisi o modelli statistici.

Eliminazione delle Righe con Valori Mancanti nei Campi Numerici

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

[9980 rows x 5 columns]
```

Il codice esegue l’operazione di rimozione delle righe con valori mancanti (NaN) dal DataFrame df. Ecco una spiegazione dettagliata:

df.dropna(subset=['NumCol1', 'NumCol2', 'NumCol3'], how='all'): Il metodo dropna() rimuove le righe contenenti valori mancanti (NaN) da un DataFrame. L’argomento subset=['NumCol1', 'NumCol2', 'NumCol3'] specifica le colonne in cui cercare i valori mancanti. L’argomento how='all' indica che una riga verrà rimossa solo se tutti i valori nelle colonne specificate sono mancanti. Quindi, il DataFrame df viene modificato e le righe con valori mancanti nelle colonne NumCol1, NumCol2 e NumCol3 vengono rimosse.

Calcolo del Totale delle Righe con Dati Mancanti

```
totale_dati_mancanti = righe_con_dati_mancanti.shape[0]
totale_dati_mancanti
```

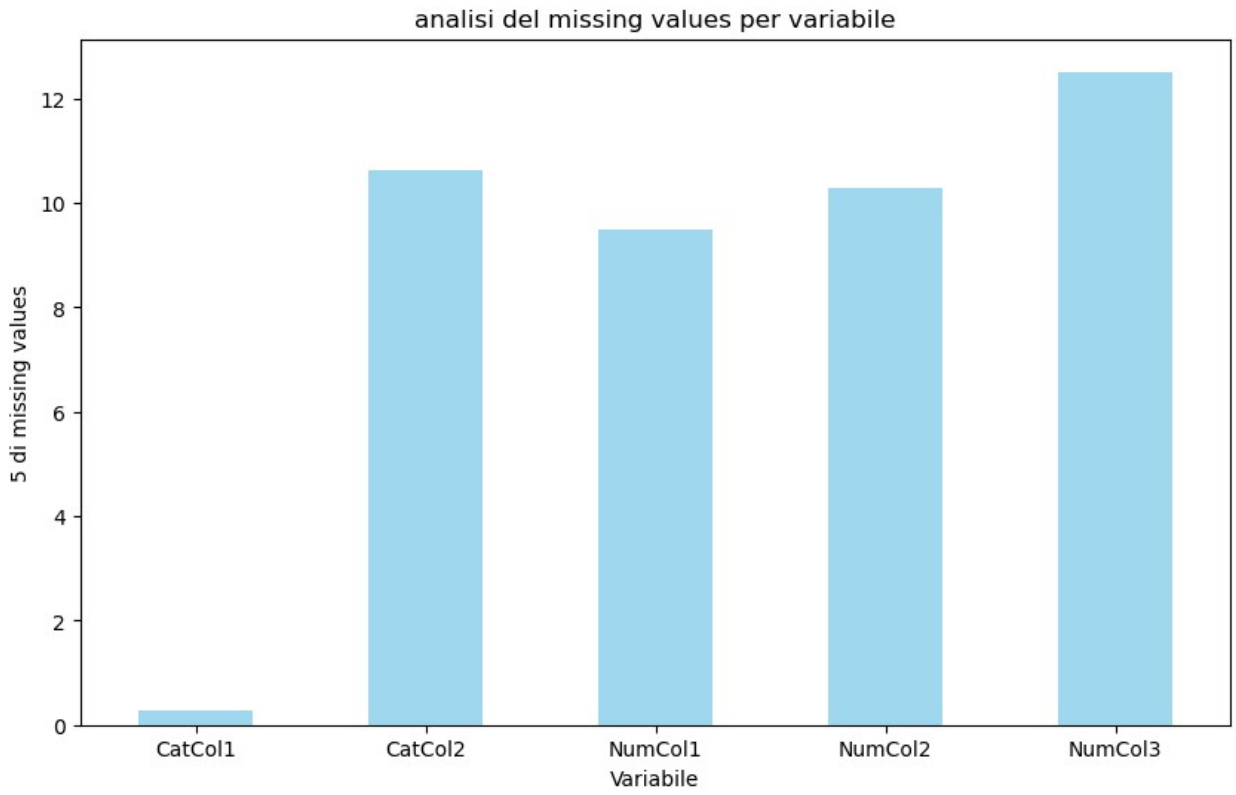
```
4
```

Il codice che hai fornito calcola il totale dei dati mancanti nelle righe selezionate del DataFrame. Ecco una spiegazione dettagliata:

righe_con_dati_mancanti.shape[0]: La proprietà shape restituisce una tupla con le dimensioni del DataFrame. In particolare, shape[0] restituisce il numero di righe nel DataFrame righe_con_dati_mancanti. Quindi, totale_dati_mancanti conterrà il numero totale di righe con dati mancanti.

Analisi Percentuale dei Missing Values per Variabile con 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()
```



Il codice che è progettato per analizzare la percentuale di valori mancanti per ciascuna variabile nel DataFrame e visualizzarli attraverso un grafico a barre utilizzando la libreria matplotlib.pyplot.

Ecco una spiegazione passo per passo del codice:

missing_percent = (df.isnull().sum()) / len(df) * 100: Questa riga calcola la percentuale di valori mancanti per ciascuna variabile nel DataFrame df. df.isnull().sum() restituisce il numero totale di valori mancanti per ogni colonna, e la divisione per len(df) fornisce la percentuale. Il risultato è memorizzato nella variabile missing_percent.

plt.figure(figsize=(10,6)): Questa riga imposta le dimensioni della figura del grafico a 10 pollici di larghezza per 6 pollici di altezza.

missing_percent.plot(kind='bar', color='skyblue', alpha=0.8): Qui viene creato il grafico a barre. kind='bar' specifica il tipo di grafico, color='skyblue' imposta il colore delle barre e alpha=0.8 regola la trasparenza delle barre.

plt.xlabel('Variabile') e plt.ylabel('5 di missing values'): Queste righe impostano le etichette degli assi x e y rispettivamente.

plt.title('analisi del missing values per variabile'): Questa riga imposta il titolo del grafico.

plt.xticks(rotation=0): Questa riga regola l'angolo di inclinazione delle etichette sull'asse x (variabili) a 0 gradi per una migliore leggibilità.

plt.show(): Questa riga mostra il grafico.

Esplorazione della Relazione tra Soddisfazione e Età: Media Condizionata di Numeric_Var per Categoria

```
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', ci=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
```

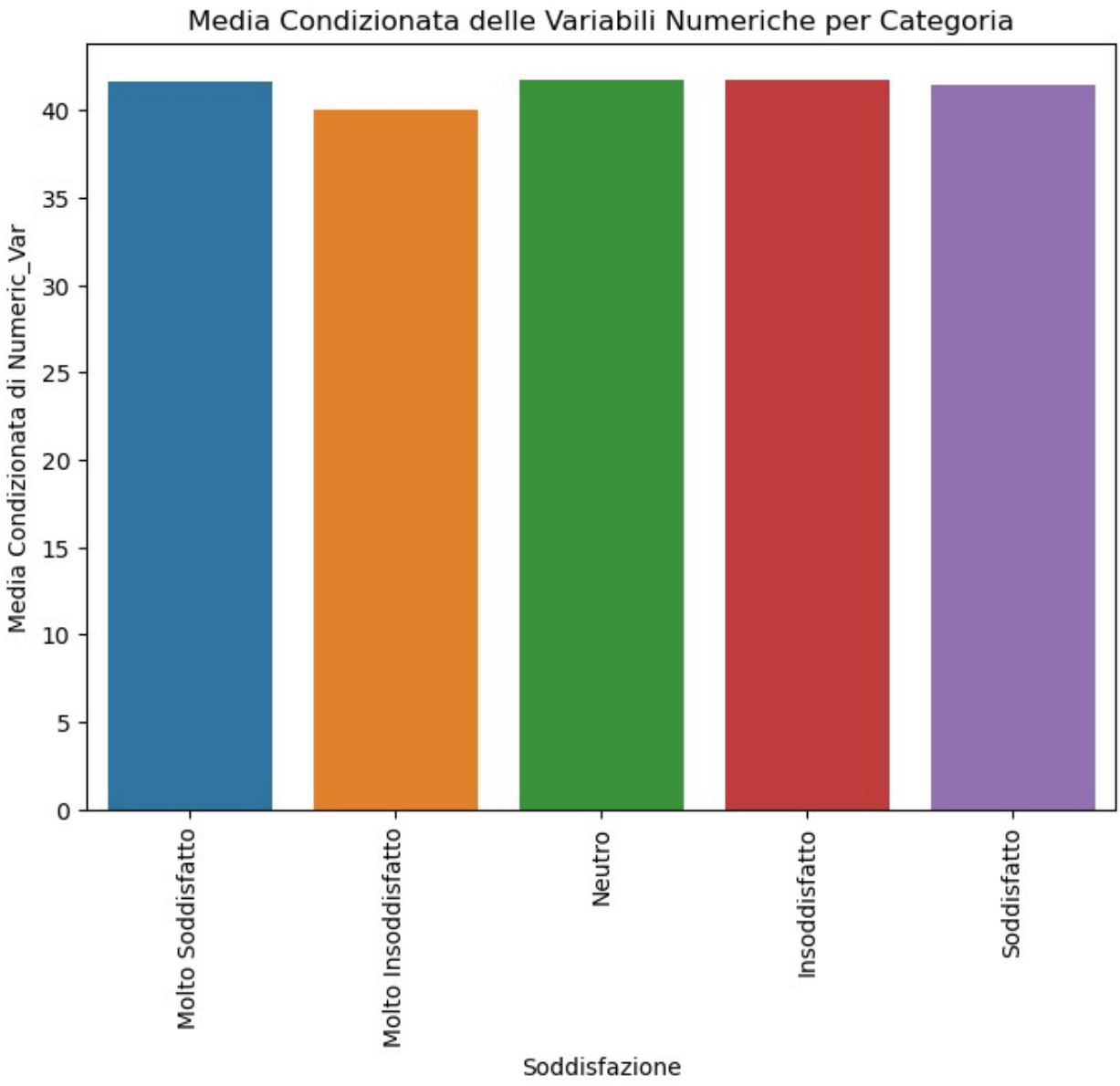
```
[500 rows x 2 columns]
     Età        Soddisfazione  Numeric_Var
0     56    Molto Soddisfatto    41.651376
1     46  Molto Insoddisfatto    40.054054
2     32               Neutro    41.747368
3     60               Neutro    41.747368
4     25  Molto Insoddisfatto    40.054054
..   ...                  ...          ...
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]

C:\Users\LucaL\AppData\Local\Temp\ipykernel_19712\3910047455.py:22:
FutureWarning:

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

  sns.barplot(data=df, x='Soddisfazione', y='Numeric_Var', ci=None)
```

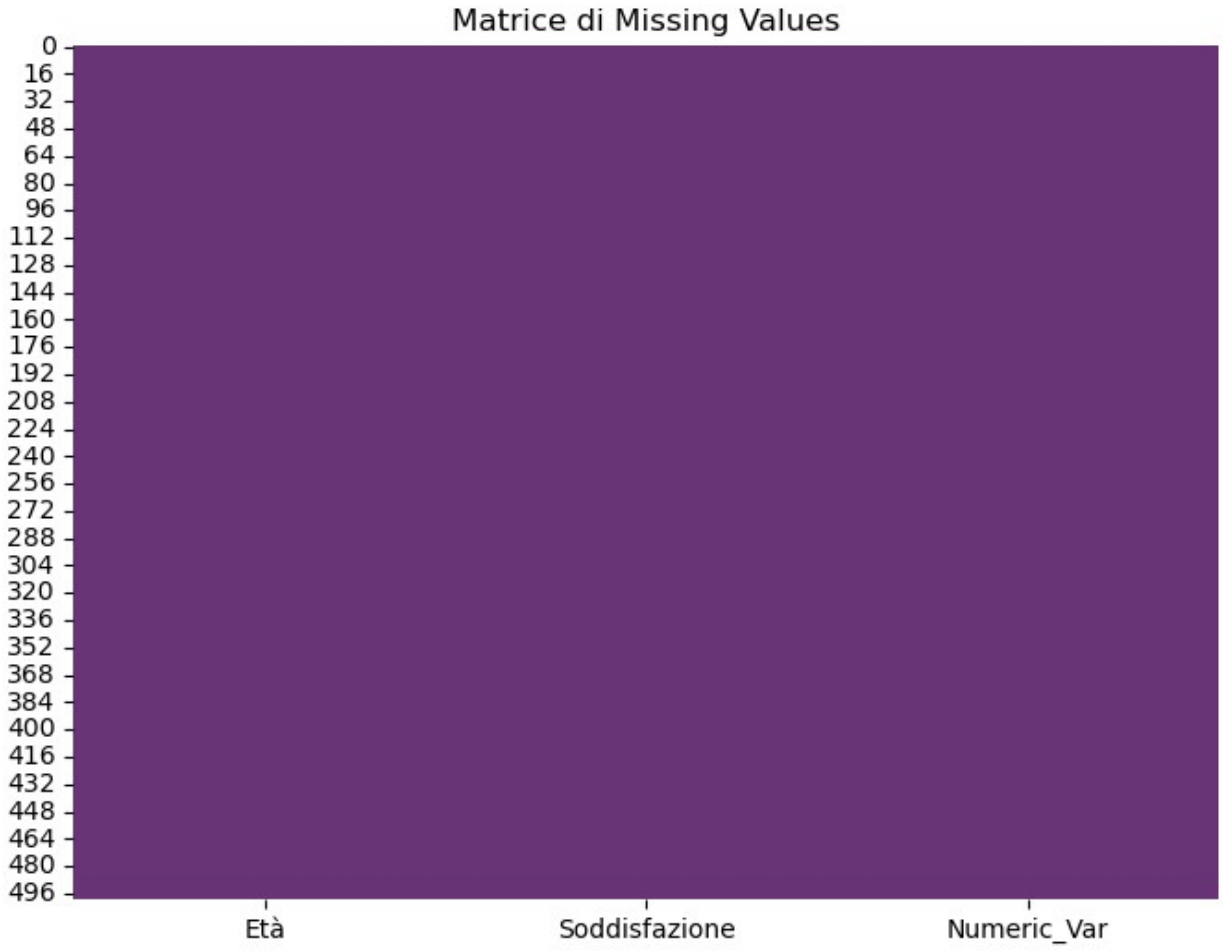


np.random.seed(42) data = { 'Età': np.random.randint(18, 65, size=500), 'Soddisfazione': np.random.choice(['Molto Soddisfatto', 'Soddisfatto', 'Neutro', 'Insoddisfatto', 'Molto Insoddisfatto'], size=500) } In questo blocco, vengono generati dati casuali per l'esplorazione. Viene fissato il seed del generatore casuale (np.random.seed(42)) per rendere i risultati riproducibili. Si creano due colonne nel DataFrame df: 'Età' con valori casuali compresi tra 18 e 65 e 'Soddisfazione' con valori casuali tra le opzioni fornite. conditional_means = df.groupby('Soddisfazione')['Età'].transform('mean') df['Numeric_Var'] = conditional_means Questo blocco calcola la media condizionata dell'età per ciascuna categoria di soddisfazione e la assegna a una nuova variabile chiamata 'Numeric_Var'. Viene utilizzato groupby per raggruppare per 'Soddisfazione' e poi transform('mean') per calcolare la media per ciascun gruppo. print(df) Qui viene stampato il DataFrame risultante dopo l'aggiunta della variabile numerica condizionale. plt.figure(figsize=(8, 6)) sns.barplot(data=df, x='Soddisfazione', y='Numeric_Var', ci=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() In questo blocco, viene creato un grafico a barre utilizzando la libreria seaborn (sns). Il grafico mostra la media condizionata della variabile numerica per ogni categoria di soddisfazione. ci=None indica che non si vogliono includere intervalli di confidenza nel grafico. L'asse x mostra le categorie di soddisfazione, l'asse y mostra la media condizionata di 'Numeric_Var' e il grafico è titolato di conseguenza. Infine, plt.show() visualizza il grafico.

Visualizzazione della Matrice dei Dati Mancanti con Heatmap Colorata

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



Creazione di una Heatmap:

python Copy code plt.figure(figsize=(8, 6)) sns.heatmap(df.isnull(), cmap='viridis', cbar=False, alpha=0.8) Questo blocco crea una heatmap utilizzando la libreria seaborn (sns). df.isnull() restituisce un DataFrame di dimensioni uguali a df, dove ogni valore è True se il valore

corrispondente in df è mancante (NaN), altrimenti False. La heatmap visualizza questi valori True e False usando i colori della mappa di colore specificata con cmap='viridis'. cbar=False specifica di non includere una barra dei colori laterale nel grafico. alpha=0.8 regola la trasparenza della heatmap.

Aggiunta del Titolo:

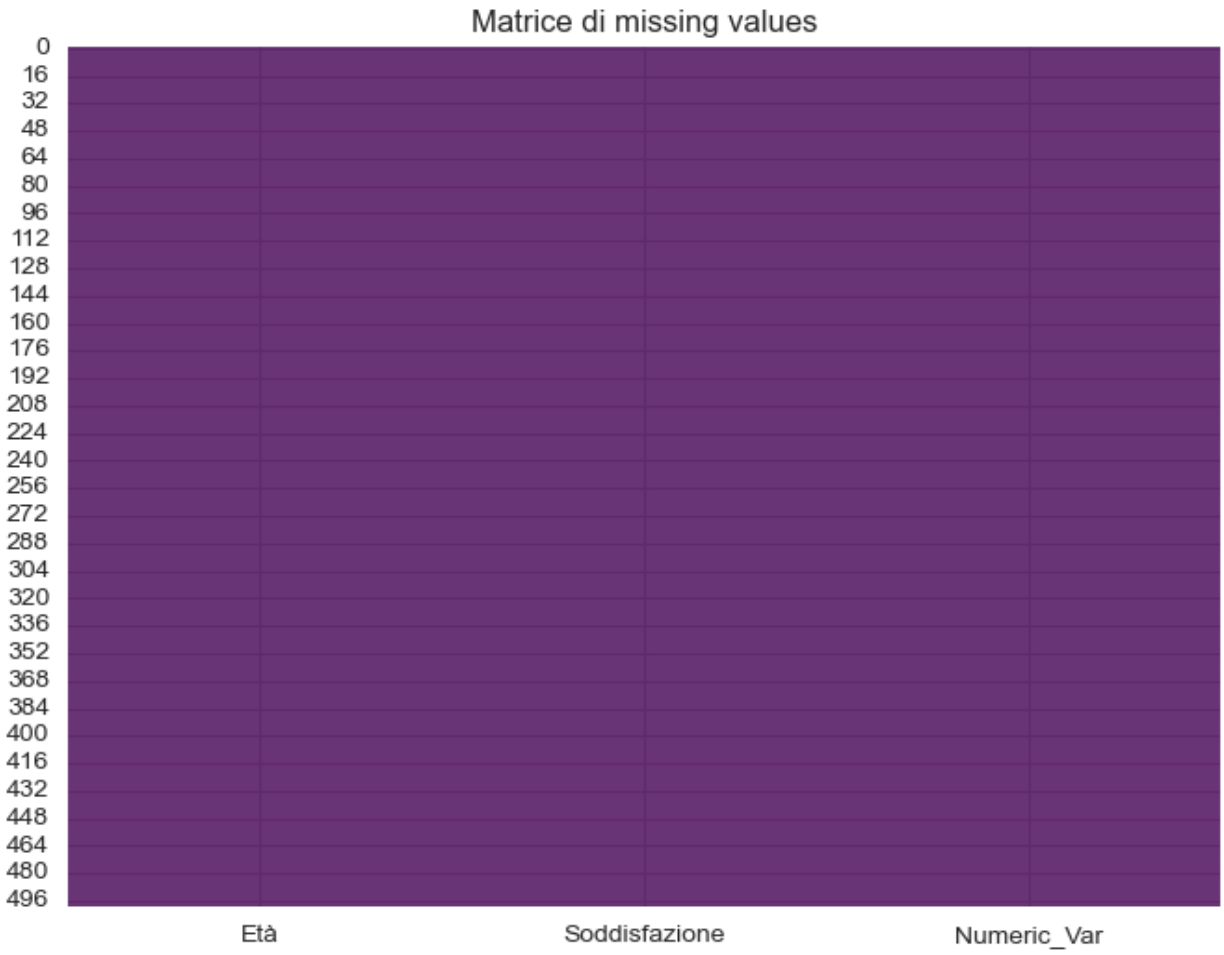
python Copy code plt.title('Matrice di Missing Values') Questo blocco aggiunge un titolo alla heatmap, indicando che si tratta della matrice dei dati mancanti.

Visualizzazione del Grafico:

python Copy code plt.show() Questo blocco visualizza la heatmap.

Visualizzazione della Matrice dei Dati Mancanti con Heatmap Colorata

```
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('Matrice di missing values')
plt.show()
```



Import data

```
 import pandas as pd
percorso_file_excel = "C:\Users\LucaL\OneDrive\Desktop\Luca Longo\
robotica\serieAnuovo.zip"
df = pd.read_excel(percorso_file_excel , sheet_name='10-11')
df

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
percorso_file_csv = "C:\Users\LucaL\OneDrive\Desktop\Luca Longo\
robotica\pokemon (1).zip"
df = pd.read_csv(percorso_file_csv)
print(df.head())
```