lavoro di gruppo missing values e import data

February 28, 2024

1 MISSING VALUES & IMPORT DATA

1.1 CREATO DA GRANIERI JOELE, TASSONE LEONARDO, RAPHAEL RODRIGO E RADISHA WARNAKULASURIYA

1.2 MISSING VALUES

1.2.1 Generazione e Visualizzazione di Dati Casuali con Pandas e NumPy

[1]:
```
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 genera un DataFrame di pandas chiamato df con 1000 righe di dati casuali. Questi dati includono:

'Età': un array di 1000 numeri interi casuali tra 18 e 70.

'Genere': un array di 1000 scelte casuali tra 'Maschio' e 'Femmina'.

'Punteggio': un array di 1000 numeri float casuali tra 0 e 100.

'Reddito': un array di 1000 numeri generati da una distribuzione normale con media 50000 e deviazione standard 15000.

La linea print(df.head()) stampa le prime 5 righe del DataFrame df. Questo è utile per avere un'anteprima dei dati senza dover stampare l'intero DataFrame.

1.2.2 Creazione e Anteprima di un DataFrame Pandas con Dati Casuali

[2]:
```
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
```

[2]:

	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 di pandas chiamato df da un elenco di dizionari. Ogni dizionario rappresenta un record con campi 'età', 'punteggio' e 'ammesso'. Alcuni dei valori in questi campi sono None, che pandas interpreta come valori mancanti o NaN (Not a Number).

1.2.3 Accesso e Manipolazione di Dati in un DataFrame Pandas

[3]:
```
df["età"]
```

[3]:
```
0    25.0
1     NaN
2    28.0
3     NaN
4    23.0
```

```
5    23.0
Name: età, dtype: float64
```

L'espressione df["età"] seleziona la colonna "età" dal DataFrame df. Questo restituirà una serie di pandas che contiene tutti i valori della colonna "età". Se ci sono valori mancanti in questa colonna, verranno rappresentati come NaN (Not a Number)

1.2.4 Identificazione di Valori Mancanti in un DataFrame Pandas

[4]:
```
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
```

[4]:

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

Il comando missing_matrix = df.isnull() crea una nuova variabile chiamata missing_matrix che è un DataFrame di stesse dimensioni di df. In missing_matrix, ogni elemento è un valore booleano che indica se l'elemento corrispondente in df è un valore mancante (True) o no (False). In questo DataFrame, True indica un valore mancante e False indica un valore non mancante.

1.2.5 Selezione di Righe con Valori Mancanti in un DataFrame Pandas

[5]:
```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

[5]:

	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 seleziona tutte le righe del DataFrame df che contengono almeno un valore mancante (NaN o None) e le assegna alla variabile righe_con_dati_mancanti.

Il comando df.isnull().any(axis=1) restituisce una serie booleana che indica se ciascuna riga ha almeno un valore mancante. Quindi, df[df.isnull().any(axis=1)] seleziona solo le righe di df per le quali il valore corrispondente nella serie booleana è True.

1.2.6 Calcolo del Numero Totale di Righe con Dati Mancanti in un DataFrame Pandas

[6]:
```
totale_dati_mancanti = righe_con_dati_mancanti.shape[0]
totale_dati_mancanti
```

[6]: 4

Il codice calcola il numero totale di righe con dati mancanti nel DataFrame df e lo assegna alla variabile totale_dati_mancanti.

La funzione shape[0] restituisce il numero di righe in un DataFrame. Quindi, righe_con_dati_mancanti.shape[0] restituisce il numero di righe nel DataFrame righe_con_dati_mancanti, che contiene solo le righe di df con almeno un valore mancante.

Il valore di totale_dati_mancanti sarà quindi il numero totale di righe con almeno un valore mancante nel DataFrame originale df.

1.2.7 Calcolo e Stampa di Dati Mancanti nel Dataset

[7]:
```
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:  4
```

Il codice stampa le righe del DataFrame df che contengono almeno un valore mancante (NaN o None), e poi stampa il numero totale di queste righe.

1.2.8 Creazione e Visualizzazione di un DataFrame con Dati Mancanti

[8]:
```
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
```

[8]:

	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 fa quanto segue:

Importa la libreria pandas con l'alias pd.

Crea un dataset, che è una lista di dizionari. Ogni dizionario rappresenta una riga di dati con quattro attributi: "nome", "età", "punteggio" e "email". Alcuni valori sono None, che rappresenta dati mancanti.

Converte il dataset in un DataFrame di pandas usando pd.DataFrame(dataset). Un DataFrame è una struttura dati bidimensionale, simile a una tabella, con righe e colonne.

Infine, il DataFrame df viene stampato.

Quindi, il codice crea un DataFrame da un dataset e lo stampa.

1.2.9 Rimozione delle Righe con Dati Mancanti dal DataFrame

[9]:
```
df1=df.dropna(inplace=False)
df1
```

[9]:

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

Il codice crea un nuovo DataFrame df1 rimuovendo tutte le righe con dati mancanti (None o NaN) dal DataFrame originale df. L'argomento inplace=False significa che la modifica non viene applicata al DataFrame originale df, ma viene restituito un nuovo DataFrame con le modifiche. Quindi, df1 sarà un DataFrame che non contiene righe con dati mancanti.

1.2.10 Creazione e Visualizzazione di una Mappa Termica dei Valori Mancanti

[10]:
```
# Crea una heatmap colorata
plt.figure(figsize=(8, 6))
#cbar serve per una barra di clore, False non lo voglio, Truese lo voglio
sns.heatmap(missing_matrix, cmap='viridis', cbar=True,alpha=0.8)
plt.title('Matrice di Missing Values')
plt.show
```

[10]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

Il codice fa quanto segue:

Crea una nuova figura con dimensioni specificate (8x6).

Utilizza la funzione heatmap della libreria seaborn (alias sns) per creare una mappa termica (heatmap) della matrice missing_matrix. La mappa termica è colorata usando il colormap 'viridis'. L'argomento cbar=True indica che vuoi visualizzare una barra di colore che mostra la scala dei colori utilizzati nella mappa termica. L'argomento alpha=0.8 imposta l'opacità dei colori nella mappa termica.

Imposta il titolo della mappa termica come 'Matrice di Missing Values'.

Infine, il comando plt.show (che dovrebbe essere plt.show()) viene utilizzato per visualizzare la figura.

Quindi, il codice crea e visualizza una mappa termica colorata della matrice missing_matrix, che rappresenta i valori mancanti nel DataFrame.

1.2.11 Creazione di DataFrame con Dati Specificati e DataFrame Vuoto

[11]:
```
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
```

[11]:

	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 fa quanto segue:

Importa le librerie necessarie: pandas (come pd), seaborn (come sns), numpy (come np) e matplotlib.pyplot (come plt).

Crea un dizionario chiamato data con tre chiavi: 'Variable1', 'Variable2' e 'Missing_Column'. Ogni chiave ha come valore una lista di cinque elementi, alcuni dei quali sono np.nan, che rappresenta un valore mancante.

Crea un DataFrame df da data utilizzando pd.DataFrame(data).

Crea un altro DataFrame vuoto df1.

Infine, stampa il DataFrame df.

Quindi, il codice crea due DataFrame, uno con i dati specificati e uno vuoto, e stampa il DataFrame con i dati.

1.2.12 Selezione delle Colonne Numeriche dal DataFrame

[12]:
```
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols
```

[12]:

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

Il codice seleziona tutte le colonne nel DataFrame df che contengono dati numerici. Questo viene fatto utilizzando il metodo select_dtypes con l'argomento include=['number']. Il risultato è un nuovo DataFrame numeric_cols che contiene solo le colonne con dati numerici dal DataFrame originale df.

1.2.13 Selezione e Stampa dei Nomi delle Colonne Numeriche dal DataFrame

[13]:
```
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols.columns
```

[13]:
```
Index(['Variable1', 'Variable2'], dtype='object')
```

Il codice fa quanto segue:

Seleziona tutte le colonne nel DataFrame df che contengono dati numerici utilizzando il metodo select_dtypes con l'argomento include=['number']. Il risultato è un nuovo DataFrame numeric_cols che contiene solo le colonne con dati numerici dal DataFrame originale df.

Infine, stampa i nomi delle colonne del DataFrame numeric_cols utilizzando l'attributo columns.

Quindi, il codice seleziona le colonne numeriche dal DataFrame df e stampa i loro nomi.

1.2.14 Sostituzione dei Valori Mancanti con la Media nelle Colonne Numeriche del DataFrame

[14]:
```
df1[numeric_cols.columns] = df[numeric_cols.columns].fillna(df[numeric_cols.
↪columns].mean())
df1

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

[14]:

	Variable1	Variable2
0	1	1.000000
1	2	2.000000
2	3	2.333333

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

Il codice fa quanto segue:

Seleziona le colonne numeriche dal DataFrame originale df utilizzando numeric_cols.columns.

Utilizza il metodo fillna per riempire i valori mancanti (NaN) nelle colonne numeriche del DataFrame df con la media (mean) dei valori esistenti in ciascuna colonna.

Assegna il risultato al DataFrame df1, creando colonne con gli stessi nomi delle colonne numeriche in df e assegnando i valori calcolati.

Infine, stampa il DataFrame df1.

Quindi, il codice crea un nuovo DataFrame df1 che ha le stesse colonne numeriche del DataFrame df, ma senza valori mancanti. I valori mancanti sono sostituiti con la media dei valori esistenti per ciascuna colonna.

1.2.15 Selezione e Stampa dei Nomi delle Colonne Categoriche dal DataFrame

[15]:
```
#trattamento dei missing values nelle variabili categoriche
categorical_cols = df.select_dtypes(exclude=['number']) #esclude colonne␣
 ↪numeriche e fa vedere solo categoriche
categorical_cols.columns
```

[15]:
```
Index(['Missing_Column'], dtype='object')
```

Il codice fa quanto segue:

Seleziona tutte le colonne nel DataFrame df che non contengono dati numerici utilizzando il metodo select_dtypes con l'argomento exclude=['number']. Questo restituisce un nuovo DataFrame categorical_cols che contiene solo le colonne con dati categorici dal DataFrame originale df.

Infine, stampa i nomi delle colonne del DataFrame categorical_cols utilizzando l'attributo columns.

Quindi, il codice seleziona le colonne categoriche dal DataFrame df e stampa i loro nomi.

1.2.16 Calcolo del Numero Totale di Valori Mancanti per Colonna nel DataFrame

[16]:
```
df.isnull().sum()
```

[16]:
```
Variable1         0
Variable2         2
Missing_Column    1
dtype: int64
```

Il codice calcola il numero totale di valori mancanti (NaN) per ogni colonna nel DataFrame df. Questo viene fatto utilizzando il metodo isnull() per identificare i valori mancanti, seguito dal metodo sum() per sommare il numero di valori mancanti per ogni

colonna. Il risultato sarà una serie in cui l'indice è il nome della colonna e il valore è il numero totale di valori mancanti in quella colonna.

1.2.17 Calcolo della Percentuale di Valori Mancanti per Colonna nel DataFrame

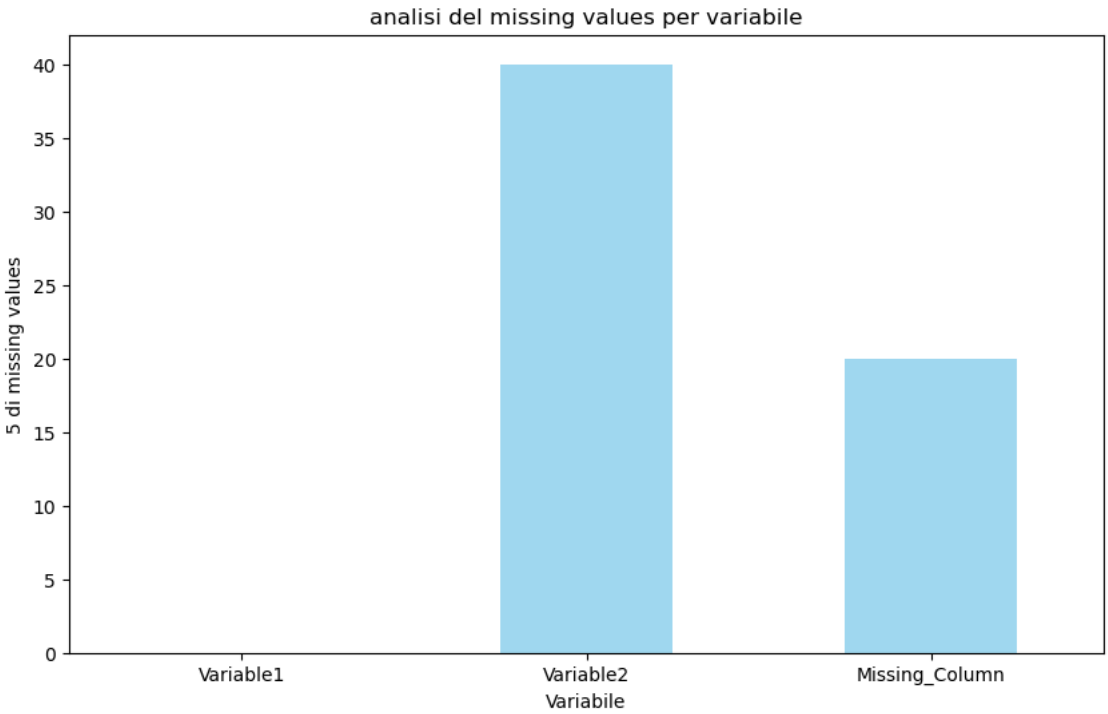
[17]:
```
missing_percent = df.isnull().sum() / len(df) * 100
missing_percent
```

[17]:
```
Variable1          0.0
Variable2         40.0
Missing_Column    20.0
dtype: float64
```

Il codice calcola la percentuale di valori mancanti (NaN) per ogni colonna nel DataFrame df. Questo viene fatto utilizzando il metodo isnull().sum() per ottenere il numero totale di valori mancanti per ogni colonna, dividendo per la lunghezza del DataFrame len(df) per ottenere la proporzione, e moltiplicando per 100 per convertire in percentuale. Il risultato sarà una serie in cui l'indice è il nome della colonna e il valore è la percentuale di valori mancanti in quella colonna.

1.2.18 Percentuale di missing values

[18]:
```
#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 fa quanto segue:

Calcola la percentuale di righe con valori mancanti (NaN) per ogni colonna nel DataFrame df. Questo viene fatto utilizzando il metodo isnull().sum() per ottenere il numero totale di valori mancanti per ogni colonna, dividendo per la lunghezza del DataFrame len(df) per ottenere la proporzione, e moltiplicando per 100 per convertire in percentuale. Il risultato è una serie in cui l'indice è il nome della colonna e il valore è la percentuale di valori mancanti in quella colonna.

Crea una nuova figura con dimensioni specificate (10x6).

Crea un grafico a barre della serie missing_percent utilizzando il metodo plot con l'argomento kind='bar'. Le barre sono colorate in 'skyblue' con un'opacità di 0.8.

Imposta le etichette degli assi x e y e il titolo del grafico.

Imposta la rotazione delle etichette dell'asse x a 0 gradi.

Infine, il comando plt.show() viene utilizzato per visualizzare la figura.

Quindi, il codice calcola la percentuale di valori mancanti per ogni colonna nel DataFrame df e crea un grafico a barre per visualizzare queste percentuali.

1.2.19 Analisi Esplorativa dei Dati Iniziale del DataFrame

[19]:
```
#informazioni sul dataset
print(df.info())

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Variable1       5 non-null      int64
 1   Variable2       3 non-null      float64
 2   Missing_Column  4 non-null      object
dtypes: float64(1), int64(1), object(1)
memory usage: 252.0+ bytes
None
       Variable1  Variable2
count   5.000000   3.000000
mean    3.000000   2.333333
std     1.581139   1.527525
min     1.000000   1.000000
25%     2.000000   1.500000
50%     3.000000   2.000000
75%     4.000000   3.000000
max     5.000000   4.000000
```

Il codice fa quanto segue:

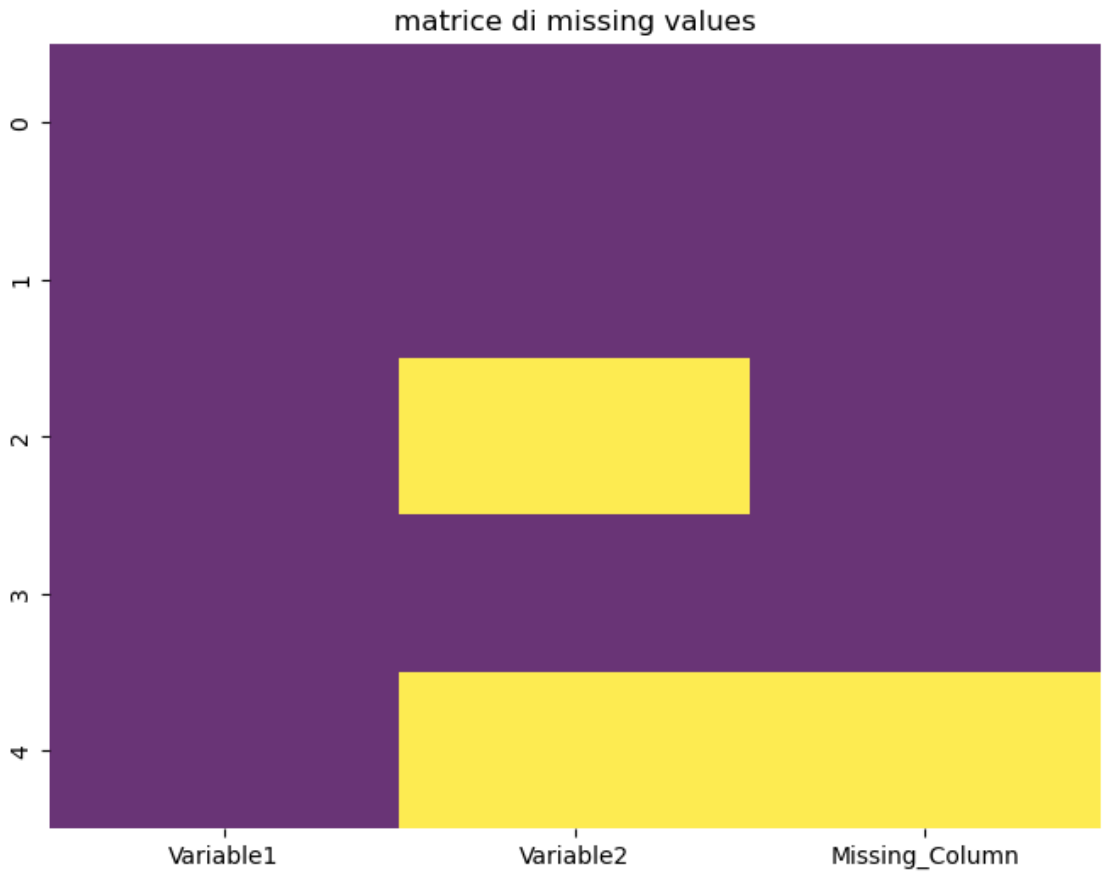
Utilizza il metodo info() sul DataFrame df per stampare informazioni sul DataFrame, come il numero di righe e colonne, i tipi di dati di ciascuna colonna, il numero di valori non nulli in ciascuna colonna e l'uso della memoria.

Utilizza il metodo describe() sul DataFrame df per stampare statistiche descrittive per ciascuna colonna, come il conteggio, la media, la deviazione standard, il minimo, il 25° percentile (Q1), la mediana (50° percentile o Q2), il 75° percentile (Q3) e il massimo.

Quindi, il codice fornisce un'analisi esplorativa dei dati (EDA) iniziale del DataFrame df, stampando informazioni generali sul DataFrame e statistiche descrittive per ciascuna colonna.

1.2.20 Visualizzazione della Posizione dei Valori Mancanti nel DataFrame tramite una Mappa Termica

[20]:
```
plt.figure(figsize=(8,6))
sns.heatmap(df.isnull(), cmap='viridis', cbar=False,alpha=0.8)
plt.title('matrice di missing values')
plt.show()
```



Il codice crea una mappa termica (heatmap) che visualizza la posizione dei valori mancanti nel DataFrame df. Le celle colorate nella mappa termica rappresentano i valori mancanti. Questo è molto utile per avere una visione rapida di dove si trovano i dati mancanti nel tuo DataFrame.

1.2.21 Generazione e Visualizzazione di un DataFrame di Dati Casuali

[21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

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

```
    'Reddito': np.random.normal(50000, 15000, size=1000) #distribuzione␣
  ↪gaussiana
}

df = pd.DataFrame(data)

# 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 fa quanto segue:

Importa le librerie necessarie: pandas, numpy, matplotlib.pyplot, seaborn e plotly.express.

Imposta un seed per il generatore di numeri casuali di numpy, in modo che i risultati siano riproducibili.

Genera un dataset di 1000 righe con quattro variabili: 'Età', 'Genere', 'Punteggio' e 'Reddito'. I dati sono generati casualmente utilizzando diverse distribuzioni: uniforme, normale e una scelta casuale tra due opzioni.

Crea un DataFrame pandas df da questo dataset.

Infine, stampa le prime cinque righe del DataFrame utilizzando il metodo head().

Quindi, il codice genera un DataFrame di dati casuali e stampa le prime cinque righe.

1.2.22 Calcolo e Stampa del Numero Totale di Valori Mancanti per Colonna nel DataFrame

[22]:
```
#gesione valori mancanti
missing_data = df.isnull().sum()
print('valori mancanti per ciascuna colonna: ')
print(missing_data)
```

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

Il codice fa quanto segue:

Utilizza il metodo isnull().sum() sul DataFrame df per calcolare il numero totale di valori mancanti (NaN) per ogni colonna. Il risultato è una serie in cui l'indice è il nome

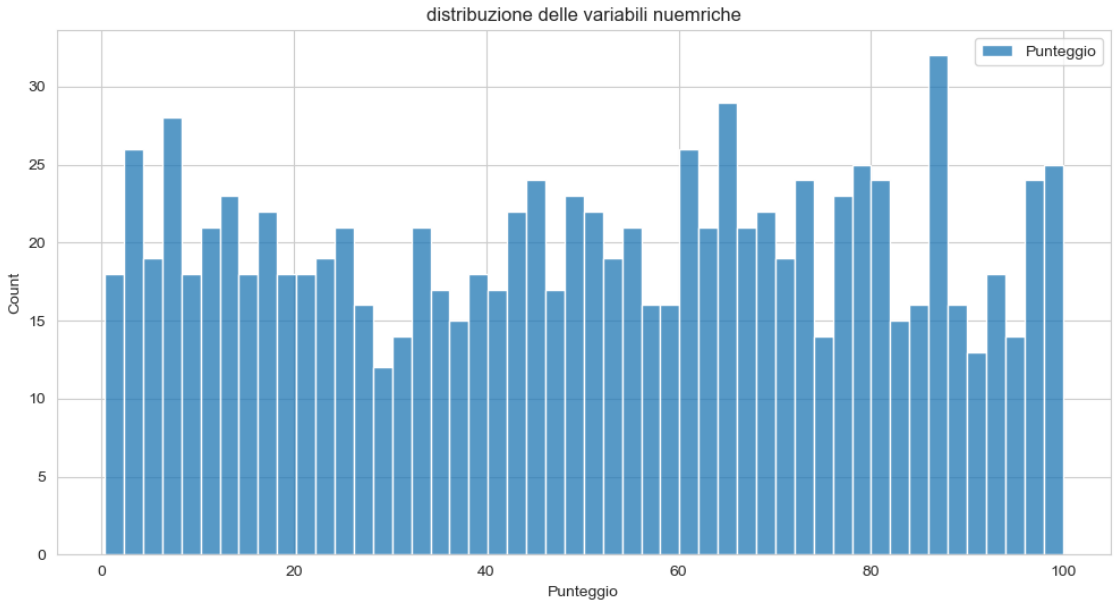
della colonna e il valore è il numero totale di valori mancanti in quella colonna.

Infine, stampa la frase 'valori mancanti per ciascuna colonna:' e la serie missing_data.

Quindi, il codice calcola e stampa il numero totale di valori mancanti per ogni colonna nel DataFrame df.

1.2.23 Creazione e Visualizzazione di un Istogramma della Distribuzione dei Valori Numerici

[23]:
```
#visualizza la distribuzione dele varibaili numeriche

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



Il codice fa quanto segue:

Crea una nuova figura con dimensioni specificate (12x6).

Imposta lo stile del grafico a 'whitegrid' utilizzando sns.set_style.

Crea un istogramma della colonna 'Punteggio' del DataFrame dfutilizzando sns.histplot. L'argomento kde=False indica che non vuoi visualizzare la curva di densità del kernel, e bins=50 indica che vuoi dividere i dati in 50 barre (o intervalli). L'argomento label='Punteggio' fornisce un'etichetta per i dati.

Visualizza la legenda del grafico con plt.legend().

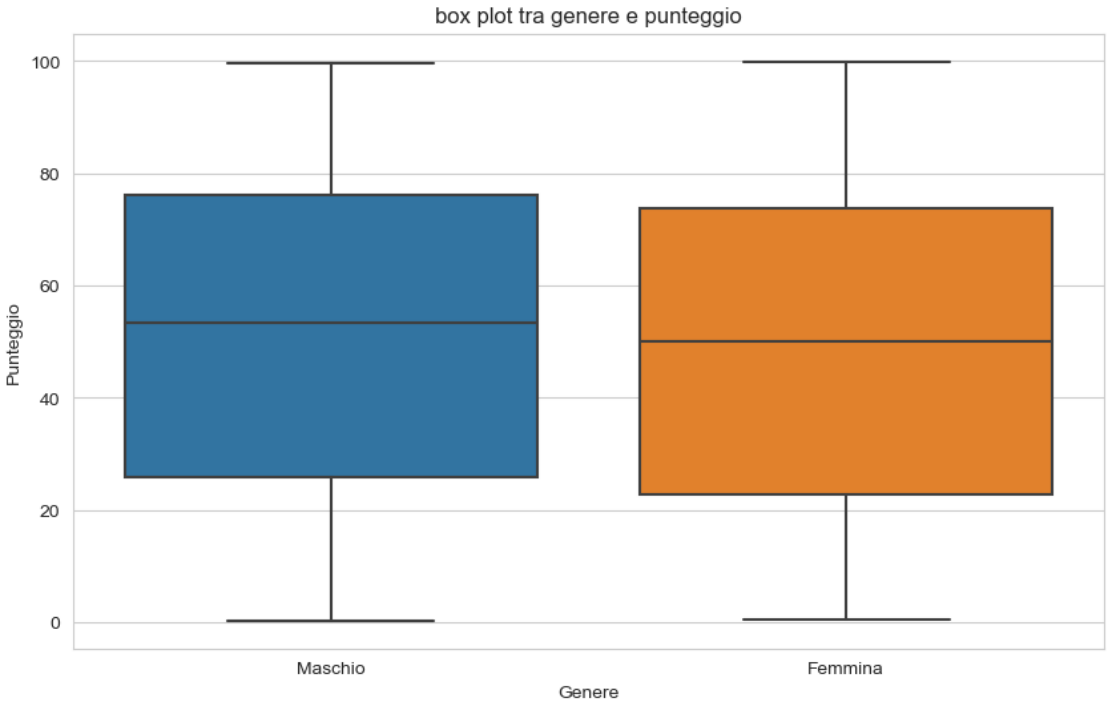
Imposta il titolo del grafico come 'distribuzione delle variabili numeriche'.

Infine, il comando plt.show() viene utilizzato per visualizzare la figura.

Quindi, il codice crea e visualizza un istogramma della distribuzione dei valori nella colonna 'Punteggio' del DataFrame df.

1.2.24 Creazione e Visualizzazione di un Box Plot della Distribuzione dei Punteggi per Genere

[24]:
```
#visualizza una box plot per una variabile numerica rispetto a un
plt.figure(figsize=(10,6))
sns.boxplot(x='Genere',y='Punteggio', data=df)
plt.title('box plot tra genere e punteggio')
plt.show()
```



Il codice fa quanto segue:

Crea una nuova figura con dimensioni specificate (10x6).

Utilizza la funzione boxplot della libreria seaborn (alias sns) per creare un box plot della colonna 'Punteggio' del DataFrame df, raggruppato per la colonna 'Genere'. Questo grafico mostra la distribuzione dei punteggi per ciascun genere.

Imposta il titolo del grafico come 'box plot tra genere e punteggio'.

Infine, il comando plt.show() viene utilizzato per visualizzare la figura.

Quindi, il codice crea e visualizza un box plot che mostra la distribuzione dei punteggi per ciascun genere nel DataFrame df.

1.2.25 Creazione e Visualizzazione di un Grafico a Dispersione Interattivo con Plotly

[25]:
```
#visualizza un grafico a dispersione interattivo utilizzando platly
import plotly.express as px
fig = px.scatter(df, x='Età', y='Reddito', color='Genere', size='Punteggio')
fig.update_layout(title='Grafico a dispersione interattivo')
fig.show()
```

Il codice fa quanto segue:

Importa la libreria plotly.express con l'alias px.

Crea un grafico a dispersione interattivo utilizzando la funzione scatter di plotly.express. Le coordinate x e y del grafico sono rispettivamente le colonne 'Età' e 'Reddito' del DataFrame df. I punti sono colorati in base alla colonna 'Genere' e la loro dimensione è determinata dalla colonna 'Punteggio'.

Aggiorna il layout del grafico per impostare il titolo come 'Grafico a dispersione interattivo'.

Infine, il comando fig.show() viene utilizzato per visualizzare il grafico.

Quindi, il codice crea e visualizza un grafico a dispersione interattivo che mostra la relazione tra 'Età' e 'Reddito', con i punti colorati in base al 'Genere' e dimensionati in base al 'Punteggio'.

1.2.26 Generazione di un DataFrame di Dati Casuali con Valori Mancanti

[26]:
```
import pandas as pd
import numpy as np

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

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

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

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

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

[26]:

	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 fa quanto segue:

Importa le librerie pandas e numpy.

Imposta un seed per il generatore di numeri casuali di numpy, in modo che i risultati siano riproducibili.

Crea un DataFrame vuoto df.

Genera dati casuali per cinque colonne nel DataFrame: 'CatCol1', 'CatCol2', 'NumCol1', 'NumCol2' e 'NumCol3'. I dati sono generati utilizzando diverse distribuzioni: scelta casuale tra opzioni specificate, distribuzione normale, distribuzione uniforme discreta e distribuzione uniforme continua.

Calcola il numero totale di valori mancanti desiderati come il 3% del numero totale di valori nel DataFrame.

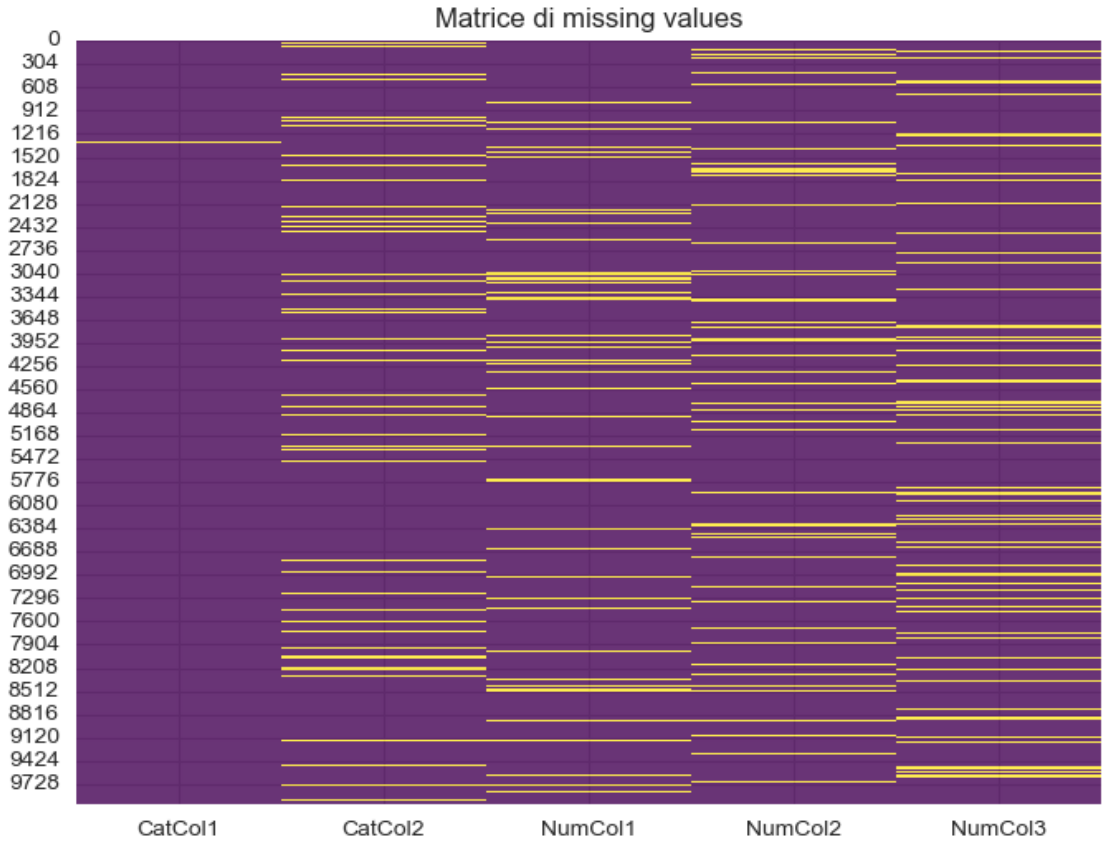
Introduce valori mancanti (NaN) casuali nel DataFrame. Per ogni colonna, sceglie un numero casuale di righe e imposta il valore in quelle righe a NaN.

Infine, stampa il DataFrame df.

Quindi, il codice genera un DataFrame di dati casuali e introduce valori mancanti casuali.

1.2.27 Creazione e Visualizzazione di una Mappa Termica dei Valori Mancanti nel DataFrame

[27]:
```
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()
```



Il codice fa quanto segue:

Crea una matrice booleana missing_matrix che indica la posizione dei valori mancanti (NaN) nel DataFrame df utilizzando il metodo isnull().

Crea una nuova figura con dimensioni specificate (8x6).

Utilizza la funzione heatmap della libreria seaborn (alias sns) per creare una mappa termica (heatmap) della matrice missing_matrix. La mappa termica è colorata usando il colormap 'viridis'. L'argomento cbar=False indica che non vuoi visualizzare una barra di colore che mostra la scala dei colori utilizzati nella mappa termica. L'argomento alpha=0.8 imposta l'opacità dei colori nella mappa termica.

Imposta il titolo della mappa termica come 'Matrice di missing values'.

Infine, il comando plt.show() viene utilizzato per visualizzare la figura.

Quindi, il codice crea e visualizza una mappa termica dei valori mancanti nel DataFrame df.

1.2.28 Generazione e Visualizzazione di un DataFrame di Dati Casuali

[28]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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

df = pd.DataFrame(data)

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

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

Il tuo codice fa quanto segue:

Importa le librerie necessarie: panda, numpy, matplotlib.pyplot e seaborn.

Imposta un seed per il generatore di numeri casuali di numpy, in modo che i risultati siano riproducibili.

Genera un dataset di 365 righe con tre variabili: 'Data', 'Vendite' e 'Prodotto'. I dati sono generati utilizzando diverse funzioni: pd.date_range genera una serie di date, np.random.randint genera numeri interi casuali e np.random.choice sceglie casualmente tra le opzioni specificate.

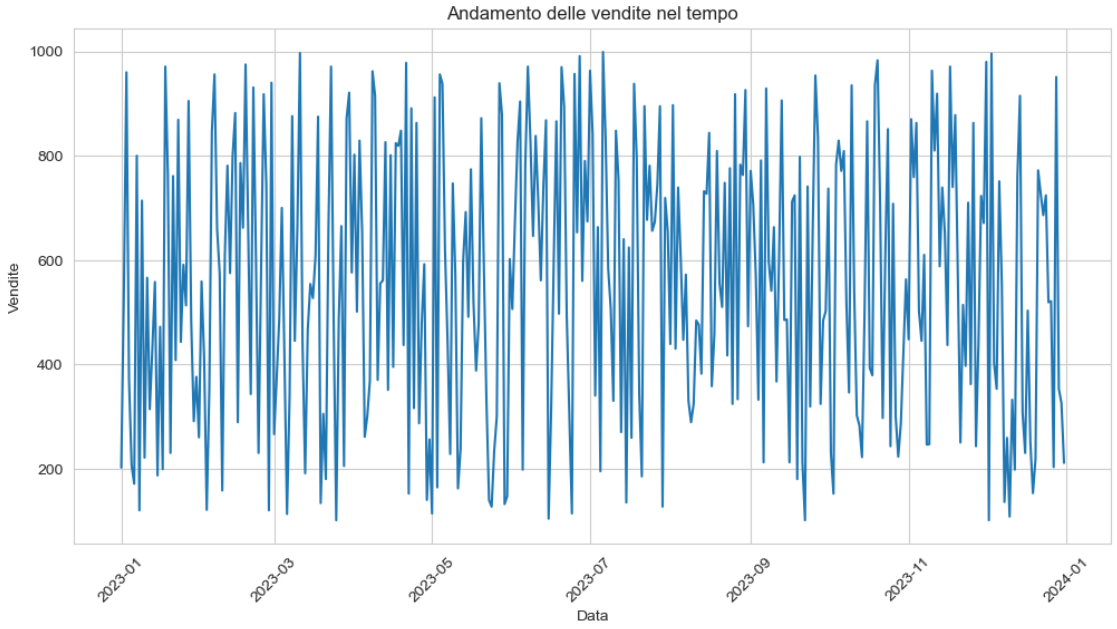
Crea un DataFrame pandas df da questo dataset.

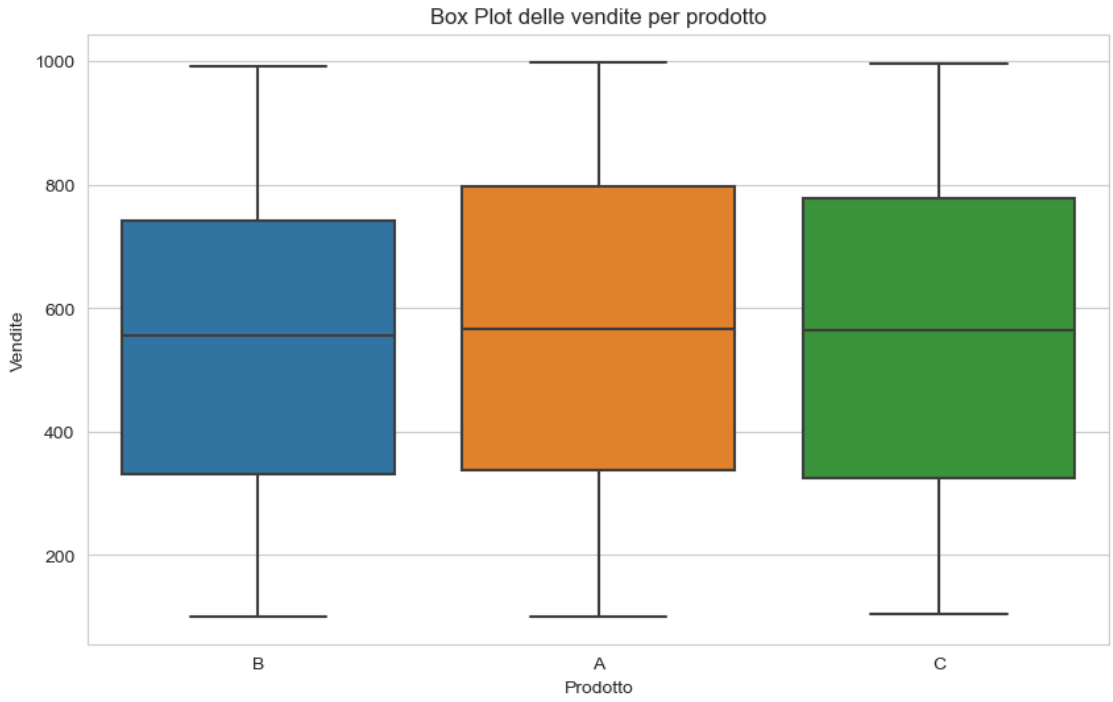
Infine, stampa le prime cinque righe del DataFrame utilizzando il metodo head().

Quindi, il tuo codice genera un DataFrame di dati casuali e stampa le prime cinque righe.

1.2.29 Creazione e Visualizzazione di un Grafico delle Vendite nel Tempo e un Box Plot delle Vendite per Prodotto

[29]:
```
# Visualizza un grafico delle vendite nel tempo
plt.figure(figsize=(12, 6))
sns.lineplot(x='Data', y='Vendite', data=df)
plt.title('Andamento delle vendite nel tempo')
plt.xlabel('Data')
plt.ylabel('Vendite')
plt.xticks(rotation=45)
plt.show()

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





Il codice fa quanto segue:

Crea un grafico delle vendite nel tempo:

Crea una nuova figura con dimensioni specificate (12x6).

Utilizza la funzione lineplot della libreria seaborn (alias sns) per creare un grafico delle vendite nel tempo. L'asse x rappresenta la data e l'asse y rappresenta le vendite.

Imposta il titolo del grafico come 'Andamento delle vendite nel tempo' e le etichette degli assi x e y come 'Data' e 'Vendite' rispettivamente.

Ruota le etichette dell'asse x di 45 gradi per una migliore leggibilità.

Infine, il comando plt.show() viene utilizzato per visualizzare il grafico.

Crea un box plot delle vendite per prodotto:

Crea una nuova figura con dimensioni specificate (10x6).

Utilizza la funzione boxplot della libreria seaborn per creare un box plot delle vendite per prodotto. L'asse x rappresenta il prodotto e l'asse y rappresenta le vendite.

Imposta il titolo del grafico come 'Box Plot delle vendite per prodotto' e le etichette degli assi x e y come 'Prodotto' e 'Vendite' rispettivamente.

Infine, il comando plt.show() viene utilizzato per visualizzare il grafico.

`</ol>`

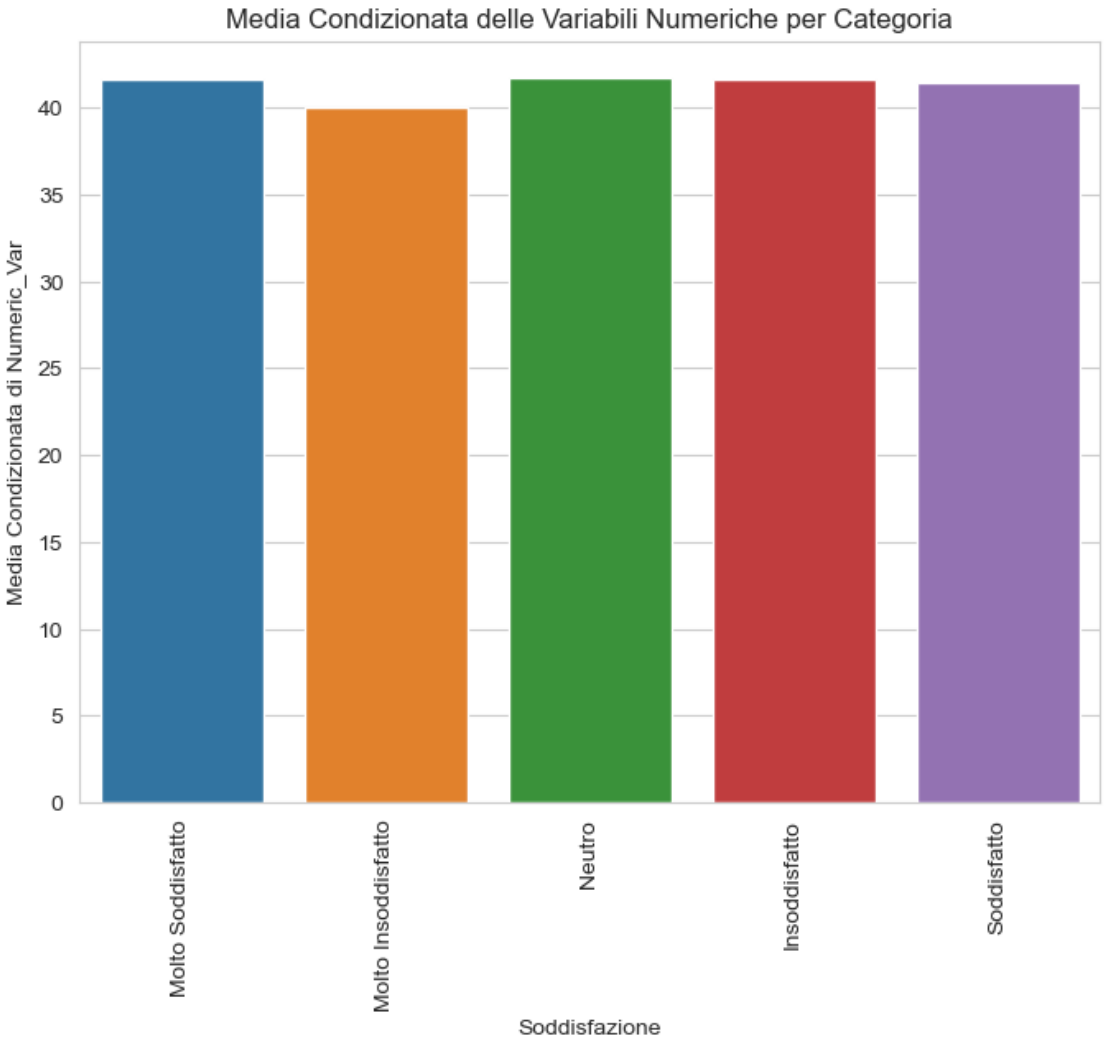
Quindi, il codice crea e visualizza due grafici: un grafico delle vendite nel tempo e un box plot delle vendite per prodotto.

1.2.30 Calcolo della Media Condizionata dell'Età per Livello di Soddisfazione e Creazione di un Grafico a Barre

[30]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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

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

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

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

plt.show()
```

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

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

[500 rows x 3 columns]
```



Il codice fa quanto segue:

Importa le librerie necessarie: pandas, numpy, matplotlib.pyplot e seaborn.

Imposta un seed per il generatore di numeri casuali di numpy, in modo che i risultati siano riproducibili.

Genera un dataset di 500 righe con due variabili: 'Età' e 'Soddisfazione'.

I dati sono generati utilizzando diverse funzioni: np.random.randint genera numeri interi casuali e np.random.choice sceglie casualmente tra le opzioni specificate.

Crea un DataFrame pandas df da questo dataset e lo stampa.

Calcola la media condizionata dell'età per ogni livello di soddisfazione utilizzando il metodo groupby e transform('mean'). Questo crea una nuova serie conditional_means che ha la stessa lunghezza del DataFrame df.

Aggiunge la serie conditional_means al DataFrame df come una nuova colonna 'Numeric_Var' e stampa il DataFrame aggiornato.

Crea una nuova figura con dimensioni specificate (8x6).

Utilizza la funzione barplot della libreria seaborn per creare un grafico a barre che mostra la media condizionata di 'Numeric_Var' per ogni livello di 'Soddisfazione'.

Imposta le etichette degli assi x e y e il titolo del grafico.

Ruota le etichette dell'asse x di 90 gradi per una migliore leggibilità.

Infine, il comando plt.show() viene utilizzato per visualizzare il grafico.

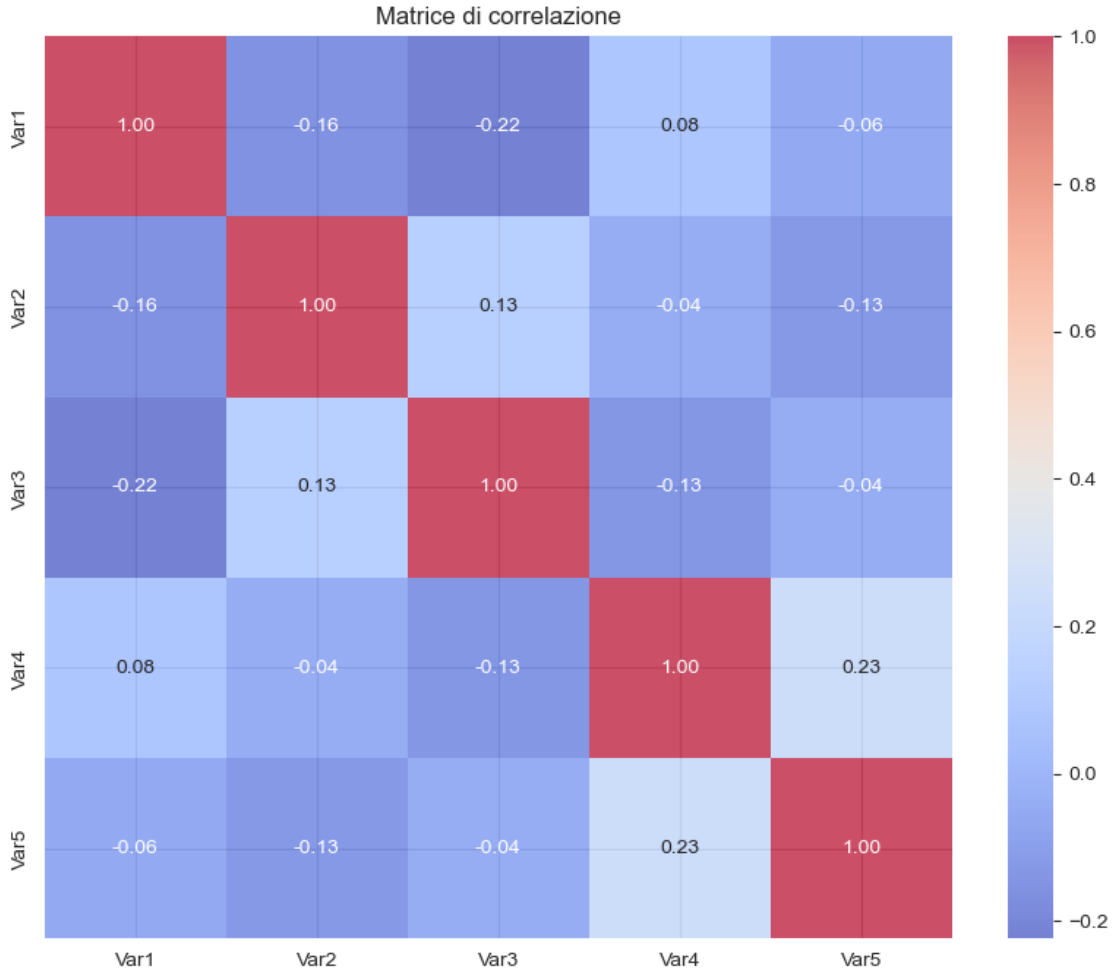
Quindi, il codice genera un DataFrame di dati casuali, calcola la media condizionata dell'età per ogni livello di soddisfazione, e crea e visualizza un grafico a barre che mostra queste medie condizionate.

1.2.31 Calcolo e Visualizzazione di una Mappa Termica della Matrice di Correlazione per un DataFrame di Dati Casuali

[31]:
```
## import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

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

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

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

#visualizza la matrice di correlazione come heatmap
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", alpha=0.
 ↪7)

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

```
C:\Users\joele\AppData\Local\Temp\ipykernel_21328\1612240019.py:14:
FutureWarning:

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



Il codice fa quanto segue:

Importa le librerie necessarie: numpy, seaborn, matplotlib.pyplot e pandas.

Imposta un seed per il generatore di numeri casuali di numpy, in modo che i risultati siano riproducibili.

Genera un DataFrame data di 100 righe con cinque variabili numeriche: 'Var1', 'Var2', 'Var3', 'Var4' e 'Var5'. I dati sono generati utilizzando una distribuzione uniforme continua tra 0 e 1.

Aggiunge due variabili categoriche al DataFrame: 'Categoria1' e 'Categoria2'. I dati sono generati scegliendo casualmente tra le opzioni specificate.

Calcola la matrice di correlazione tra tutte le variabili numeriche utilizzando il metodo corr(). Questo restituisce un nuovo DataFrame correlation_matrix in cui l'indice e le colonne sono i nomi delle variabili numeriche e ogni cella contiene il coefficiente di correlazione tra due variabili.

Crea una nuova figura con dimensioni specificate (10x8).

Utilizza la funzione heatmap della libreria seaborn per creare una mappa termica (heatmap) della matrice di correlazione. La mappa termica è colorata usando il colormap 'coolwarm', e ogni cella è annotata con il coefficiente di correlazione corrispondente, formattato come un numero decimale con due cifre decimali.

Imposta il titolo della mappa termica come 'Matrice di correlazione'.

Infine, il comando plt.show() viene utilizzato per visualizzare la figura.

Quindi, il codice genera un DataFrame di dati casuali, calcola la matrice di correlazione tra le variabili numeriche e crea e visualizza una mappa termica di questa matrice di correlazione.

1.2.32 Identificazione e Conteggio delle Righe con Dati Mancanti nel DataFrame

[32]:
```
#identificazione delle righe con dati mancati
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
len(righe_con_dati_mancanti)
```

[32]:
```
0
```

Il codice fa quanto segue:

Utilizza il metodo isnull().any(axis=1) sul DataFrame df per creare una serie booleana che indica se c'è un valore mancante (NaN) in qualsiasi colonna di ciascuna riga.

Seleziona le righe del DataFrame df che hanno almeno un valore mancante, utilizzando la serie booleana come indice. Questo restituisce un nuovo DataFrame righe_con_dati_mancanti che contiene solo le righe con valori mancanti.

Infine, calcola e stampa il numero di righe nel DataFrame righe_con_dati_mancanti utilizzando la funzione len().

Quindi, il codice identifica le righe con dati mancanti nel DataFrame df e stampa il loro numero.

1.2.33 Percentuale di valori mancanti per colonna

[33]:
```
missing_percent = (df.isnull().sum() / len(df) * 100)
missing_percent
```

[33]:
```
Età             0.0
Soddisfazione   0.0
Numeric_Var     0.0
dtype: float64
```

Questo codice calcola la percentuale di valori mancanti in ciascuna colonna del DataFrame df:

df.isnull() restituisce un DataFrame dello stesso formato di df, ma con valori True dove i valori originali erano mancanti (NaN) e False dove i valori erano presenti.

.sum() somma i valori True (considerati come 1) in ciascuna colonna, restituendo il numero totale di valori mancanti per colonna.

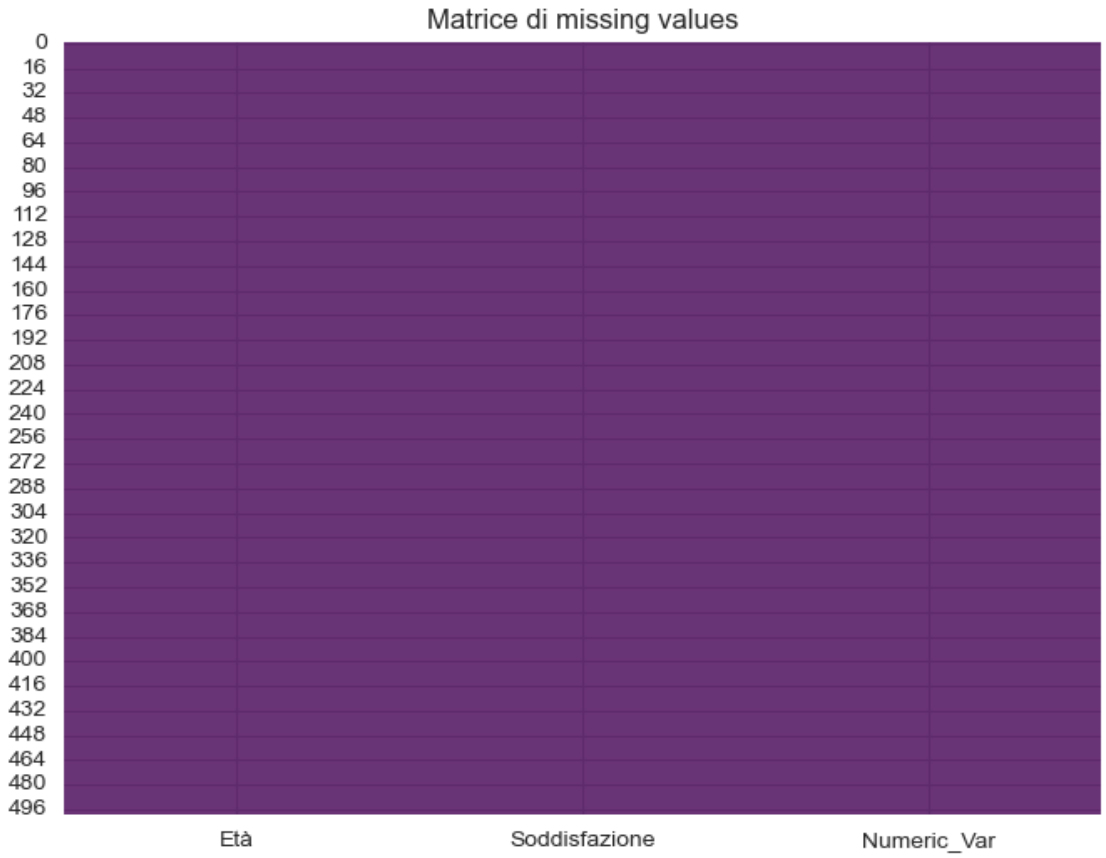
Dividendo per len(df) si ottiene la proporzione di valori mancanti per colonna, poiché len(df) restituisce il numero totale di righe nel DataFrame.

Moltiplicando per 100, si converte la proporzione in percentuale.

Il risultato, missing_percent, è una serie pandas in cui l'indice è l'elenco delle colonne di df e i valori sono le percentuali di valori mancanti per ciascuna colonna.

1.2.34 Visualizzazione dei Valori Mancanti con Heatmap

[34]:
```
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()
```



Il codice fa quanto segue:

Crea una matrice di valori mancanti: Il comando missing_matrix = df.isnull() crea una nuova matrice chiamata missing_matrix. Questa matrice ha la stessa forma del DataFrame originale df, ma contiene valori True dove i valori originali erano mancanti (NaN) e False dove i valori erano presenti.

Crea una heatmap dei valori mancanti: Il comando plt.figure(figsize=(8,6)) crea una nuova figura con dimensioni specificate (8x6). Il comando sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8) utilizza la funzione heatmap della libreria seaborn (alias sns) per creare una heatmap dei valori mancanti. L'asse x rappresenta le colonne del DataFrame e l'asse y rappresenta le righe. I valori mancanti sono rappresentati con un colore e i valori non mancanti con un altro colore. Il comando plt.title('Matrice di missing values') imposta il titolo della heatmap come 'Matrice di missing values'. Infine, il comando plt.show() viene utilizzato per visualizzare la heatmap.

Quindi, il codice crea e visualizza una heatmap dei valori mancanti nel DataFrame df. Questo può essere molto utile per capire dove si concentrano i dati mancanti.

1.3 IMPORT DATA

1.3.1 Caricamento e Visualizzazione dei Dati da un Foglio di Lavoro Excel in Python

[35]:
```
import pandas as pd

percorso_file_excel = "C:\\Users\\joele\\OneDrive\\Desktop\\dati␣
 ↪robotica\\serieA.xlsx"
df = pd.read_excel(percorso_file_excel , sheet_name='10-11')
df
```

[35]:

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

	Goals suffered	Difference goals
0	24	41
1	42	27
2	39	20
3	43	22
4	39	16
5	52	7
6	47	10
7	63	-5
8	44	5
9	47	-2
10	40	-2
11	47	-8
12	52	-12
13	51	-7

```
14              50              -12
15              52              -17
16              66              -20
17              49              -16
18              52              -18
19              56              -29
```

il codice sta facendo quanto segue:

Importa la libreria pandas: La prima riga del codice importa la libreria pandas e la rinomina come pd. Pandas è una libreria di Python che fornisce strutture dati e funzionalità per manipolare e analizzare i dati.

Specifica il percorso del file Excel: La variabile percorso_file_excel viene impostata con il percorso del file Excel che desideri leggere. In questo caso, il file si trova nella cartella 'dati robotica' sul tuo desktop.

Legge il file Excel: La funzione pd.read_excel() viene utilizzata per leggere il file Excel specificato. Il parametro sheet_name='10-11' indica che desideri leggere il foglio di lavoro chiamato '10-11' nel file Excel.

Visualizza il DataFrame: L'ultima riga del codice, df, stampa il contenuto del DataFrame df. Un DataFrame è una struttura dati bidimensionale, simile a una tabella, che può contenere dati di vari tipi (numerici, stringhe, booleani, ecc.) e consente operazioni di manipolazione dei dati come quelle in SQL e Excel.

In sintesi, il codice legge un foglio di lavoro specifico da un file Excel e lo carica in un DataFrame di pandas, che viene poi visualizzato.

1.3.2 Caricamento e Visualizzazione dei Dati da un File CSV in Python

[36]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
percorso_file_csv = "C:\\Users\\joele\\OneDrive\\Desktop\\dati␣
 ↪robotica\\pokemons.csv"
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    None  39
4   5  charmeleon  ordinary  generation-i   charmander   fire    None  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…
```

il codice fa quanto segue:

Importa le librerie necessarie: Le prime righe del codice importano le librerie necessarie: pandas (rinominata come pd), numpy (rinominata come np), matplotlib.pyplot (rinominata come plt) e seaborn (rinominata come sns). Queste librerie forniscono funzionalità per manipolare e analizzare i dati, così come per la visualizzazione dei dati.

Specifica il percorso del file CSV: La variabile percorso_file_csv viene impostata con il percorso del file CSV che desideri leggere. In questo caso, il file si trova nella cartella 'dati robotica' sul tuo desktop.

Legge il file CSV: La funzione pd.read_csv() viene utilizzata per leggere il file CSV specificato. Questa funzione legge il file CSV e lo converte in un DataFrame di pandas.

Visualizza il DataFrame: L'ultima riga del codice, print(df.head()), stampa le prime 5 righe del DataFrame df. Un DataFrame è una struttura dati bidimensionale, simile a una tabella, che può contenere dati di vari tipi (numerici, stringhe, booleani, ecc.) e consente operazioni di manipolazione dei dati.

In sintesi, il codice legge un file CSV specifico e lo carica in un DataFrame di pandas, che viene poi visualizzato stampando le prime 5 righe.