Missing Values - Import Datti

Componenti del gruppo:

Justin Cadena
Francesco Miraglia
Zhou Zencheng

Indice:

1) Missing Values, 3
2) Import dati, 21

1 Missing Values

1.1 Creazione di un DataFrame con dati mancanti rappresentati da Nan e None

[2]:
```
import pandas as pd

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

Commento: Il codice crea un DataFrame utilizzando la libreria Pandas in Python. Il DataFrame contiene dati su età, punteggio e ammissione. Successivamente, il DataFrame viene stampato a schermo.

[3]:
```
#Seleziona solo le colonne 'punteggio' e 'ammesso' dal DataFrame
df[["punteggio", "ammesso"]]
```

[3]:

	punteggio	ammesso
0	90.0	1.0
1	85.0	0.0
2	NaN	1.0
3	75.0	1.0
4	NaN	NaN
5	77.0	NaN

[4]:
```
# Trova le righe contenenti dati mancanti nel DataFrame
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

[4]:

	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
```

[5]:
```
# Calcola il totale delle righe con dati mancanti nel DataFrame
totale_dati_mancanti = righe_con_dati_mancanti.shape[0]
totale_dati_mancanti
```

[5]:
```
5
```

[6]:
```
# Stampa le righe con dati mancanti e il totale dei dati mancanti
print("Righe con dati mancanti:")
print(righe_con_dati_mancanti)
print("Totale dati mancanti:", totale_dati_mancanti)
```

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

1.2 Creazione di un DataFrame con punteggi

[7]:
```
import pandas as pd

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"},
]

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

[7]:
```
    età  punteggio  ammesso
0  25.0       90.0      1.0
```

Commento: Il codice Python utilizza la libreria pandas per gestire un insieme di dati. Questi dati sono rappresentati come una lista di dizionari, dove ogni dizionario rappresenta una persona con attributi come nome, età, punteggio ed email. Il codice poi elimina tutte le righe che contengono valori mancanti.

[8]:
```
# Elimina le righe con dati mancanti dal DataFrame originale
df.dropna(inplace=True)
```

```
df
```

[8]:
```
    età  punteggio  ammesso
0  25.0       90.0      1.0
```

[9]:
```
# Elimina le righe con dati mancanti dal DataFrame originale
df.dropna(inplace=True)
df
```

[9]:
```
    età  punteggio  ammesso
0  25.0       90.0      1.0
```

1.3 Creazione di un DataFrame con dati mancanti e calcolo delle statistiche

[10]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

data = {
    'Variable1': [1, 2, 3, 4, 5],
    'Variable2': [1, 2, np.nan, 4, np.nan],
    'Missing_Column': ['A', 'B', 'A', 'C', np.nan]
}

df = pd.DataFrame(data)
df1=pd.DataFrame()
df
```

[10]:
```
   Variable1  Variable2 Missing_Column
0          1        1.0              A
1          2        2.0              B
2          3        NaN              A
3          4        4.0              C
4          5        NaN            NaN
```

1.4 Selezione delle colonne numeriche

[11]:
```
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols.columns
```

[11]:
```
Index(['Variable1', 'Variable2'], dtype='object')
```

1.5 Sostituzione dei valori mancanti con le medie delle colonne numeriche

[12]:
```
df1[numeric_cols.columns] = df[numeric_cols.columns].fillna(df[numeric_cols.
↪columns].mean())
df1
```

[12]:
```
   Variable1  Variable2
0          1   1.000000
1          2   2.000000
2          3   2.333333
3          4   4.000000
4          5   2.333333
```

[13]:
```
categorical_cols = df.select_dtypes(exclude=['number'])
categorical_cols.columns
```

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

1.6 Sostituzione dei valori mancanti con le mode delle colonne categoriche

[14]:
```
df1[categorical_cols.columns] = df[categorical_cols.columns].
↪fillna(df[categorical_cols.columns].mode().iloc[0])
df1
```

[14]:
```
   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
```

1.7 Stampa dei DataFrame: Primo con i valori mancanti e secondo con i valori mancanti sostituiti

[15]:
```
print(f"il primo con i valori mancanti \n{df} \ne il secondo con i missing␣
↪values sostituiti \n{df1}")
```

```
il primo con i valori mancanti
   Variable1  Variable2 Missing_Column
0          1        1.0              A
1          2        2.0              B
2          3        NaN              A
3          4        4.0              C
4          5        NaN            NaN
e il secondo con i missing values sostituiti
   Variable1  Variable2 Missing_Column
0          1   1.000000              A
1          2   2.000000              B
```

```
2           3   2.333333              A
3           4   4.000000              C
4           5   2.333333              A
```

1.8 Creazione di un DataFrame con Dati di Esempio

[16]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

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

[16]:

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

1.9 Conteggio dei valori mancanti per ciascuna colonna nel dataFrame

[17]:
```
df.isnull().sum()
```

[17]:
```
Feature1    1
Feature2    2
Feature3    1
dtype: int64
```

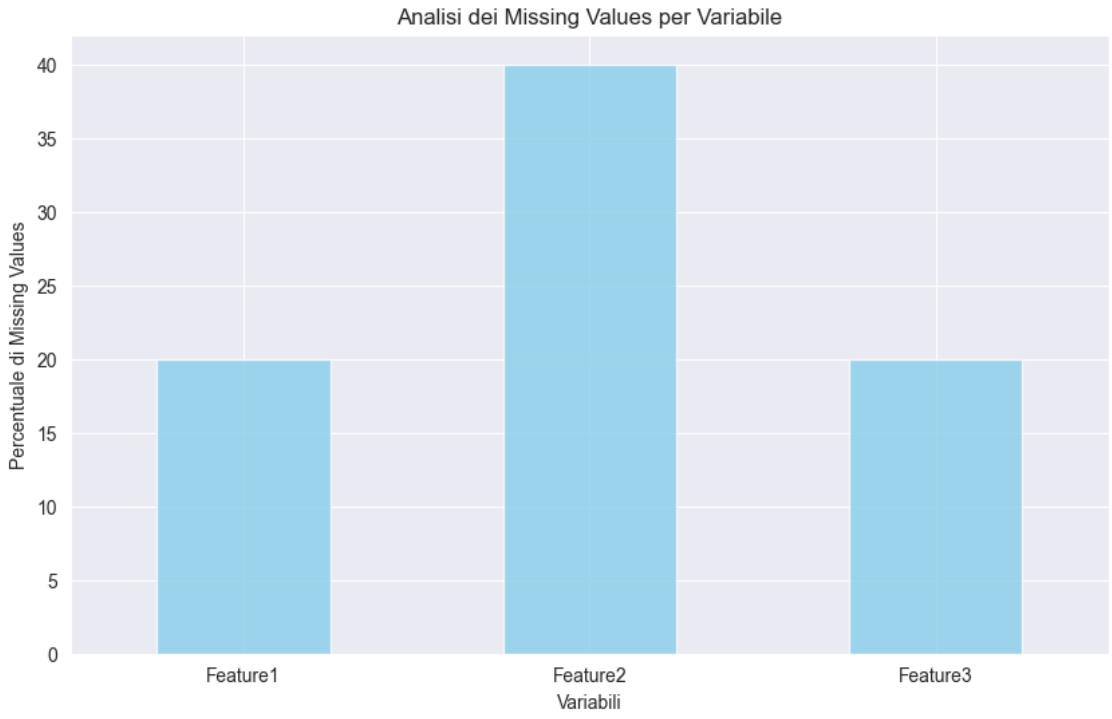
1.10 Percentuale di valori mancanti per ciascuna colonna nel dataFrame

[18]:
```
missing_percent = (df.isnull().sum() / len(df)) * 100
missing_percent
```

[18]:
```
Feature1    20.0
Feature2    40.0
Feature3    20.0
dtype: float64
```

1.11 Grafico del Dataframe

[19]:
```
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()
```



1.12 Funzione per il trattamento dei valori mancanti in variabili numeriche e categoriche

[20]:
```
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], # Aggiunge valori mancanti (np.nan)
    'Missing_Column': ['A', 'B', 'A', 'C', np.nan]
}
```

```
# Crea un DataFrame a partire dai dati
df = pd.DataFrame(data)
df1=pd.DataFrame() # Crea un DataFrame vuoto per memorizzare i dati con valori␣
 ↪mancanti


def missingvalues_sub(df):
    # Trattamento dei missing values nelle variabili numeriche e categoriche

    # Seleziona le colonne numeriche e categoriche
    numeric_cols = df.select_dtypes(include=['number'])
    categorical_cols = df.select_dtypes(exclude=['number'])

     # Sostituisce i valori mancanti nelle colonne numeriche con la media dei␣
 ↪valori non mancanti
    df1[numeric_cols.columns] = df[numeric_cols.columns].fillna(df[numeric_cols.
 ↪columns].mean())

     #Sostituisce i valori mancanti nelle colonne categoriche con la moda dei␣
 ↪valori non mancanti
    df1[categorical_cols.columns] = df[categorical_cols.columns].
 ↪fillna(df[categorical_cols.columns].mode().iloc[0])
    return df1


def main ():

   # Chiama la funzione per il trattamento dei valori mancanti
    df1=missingvalues_sub(df)

# Stampa i DataFrame originale e con i valori mancanti sostituiti
    print(f"il primo con i valori mancanti \n{df} \ne il secondo con i missing␣
 ↪values sostituiti \n{df1}")

#  Esegue la funzione principale quando il programma viene avviato direttamente
if __name__ == "__main__":
    main()
```

```
il primo con i valori mancanti
   Variable1  Variable2 Missing_Column
0          1        1.0              A
1          2        2.0              B
2          3        NaN              A
3          4        4.0              C
4          5        NaN            NaN
e il secondo con i missing values sostituiti
```

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

1.13 Individua i valori mancanti

[21]:
```
df.isnull()
```

[21]:
```
   Variable1  Variable2  Missing_Column
0      False      False           False
1      False      False           False
2      False       True           False
3      False      False           False
4      False       True            True
```

1.14 Creazione di una matrice di valori Mancanti in un DataFrame

[22]:
```
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
```

[22]:
```
   Feature1  Feature2  Feature3
0     False      True     False
1     False     False      True
2      True     False     False
3     False     False     False
4     False      True     False
```

1.15 Grafico della matrice

[23]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
plt.title('Matrice di Missing Values')
plt.show()
```

1.16 Creazione di un DataFrame con dati Casuali su Età, Genere, Punteggio e Reddito

[25]:
```
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'+
np.random.seed(2)
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)
}

df = pd.DataFrame(data)

# Visualizza le prime righe del dataset
print(df.head(21))
```

```
    Età   Genere  Punteggio       Reddito
0    58  Maschio  93.309731  55174.034340
1    33  Femmina  97.279382  65873.059029
2    63  Femmina  91.185842  63246.553249
3    26  Femmina  75.926276  44534.875858
4    40  Maschio  25.156395  73444.267270
5    61  Femmina  90.055564  48451.939402
6    36  Femmina  29.717079  44579.517216
7    29  Femmina  87.762886  74639.606864
8    58  Femmina   4.139801  84279.892767
9    25  Femmina   5.641115  52083.863707
10   52  Maschio  80.315899  58188.649042
11   67  Maschio  10.670863  40301.012748
12   49  Maschio  43.920719  58292.619116
13   29  Femmina  34.315554  54842.947703
14   39  Maschio  27.790752  53270.120207
15   65  Maschio  36.205126  78821.228153
16   49  Maschio  48.566180  59639.075018
17   44  Femmina  83.643168  39339.223303
18   38  Maschio  61.718371  40687.283872
19   55  Femmina  90.736827  66795.123408
20   57  Maschio  83.670954  66695.930851
```

1.17 Informazione e descrizione del Dataframe

[26]:
```
print(df.info())

print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Età        1000 non-null   int32
 1   Genere     1000 non-null   object
 2   Punteggio  1000 non-null   float64
 3   Reddito    1000 non-null   float64
```

```
dtypes: float64(2), int32(1), object(1)
memory usage: 27.5+ KB
None
                Età    Punteggio       Reddito
count  1000.000000  1000.000000   1000.000000
mean     44.205000    48.687071  50036.084395
std      14.986847    29.617200  15027.142896
min      18.000000     0.090182   6017.070033
25%      31.000000    22.373740  39577.758808
50%      44.000000    47.030664  50994.854630
75%      58.000000    75.439618  60933.234680
max      69.000000    99.713537  96435.848804
```

1.18 Valori mancanti per ciascuna colonna

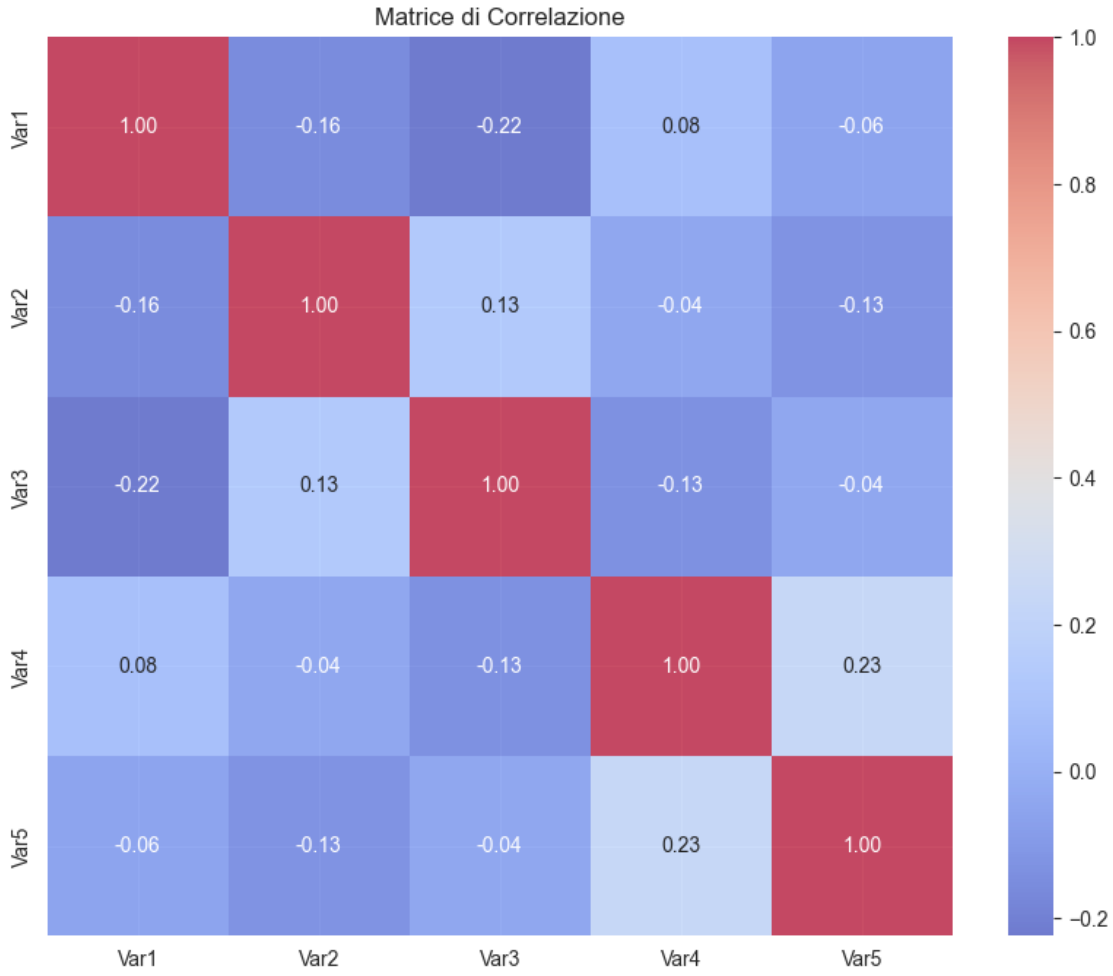
[27]:
```
missing_data = df.isnull().sum()
print("Valori mancanti per ciascuna colonna:")
print(missing_data)
```

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

1.19 Grafico matrice di correlazione

[28]:
```
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'])

# Aggiunge 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 solo delle colonne numeriche
correlation_matrix = data.corr(numeric_only=True)

# Visualizza la matrice di correlazione in una 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()
```



1.20 Generazione di un DataFrame con dati Casuali e introduzione di valori mancanti casuali

[29]:
```
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
```

[29]:

	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]

1.21 Conteggio delle righe con dati mancanti

[30]:
```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
len(righe_con_dati_mancanti)
```

[30]: 3648

1.22 Percentuale di valori mancanti di ogni colonna del Dataframe

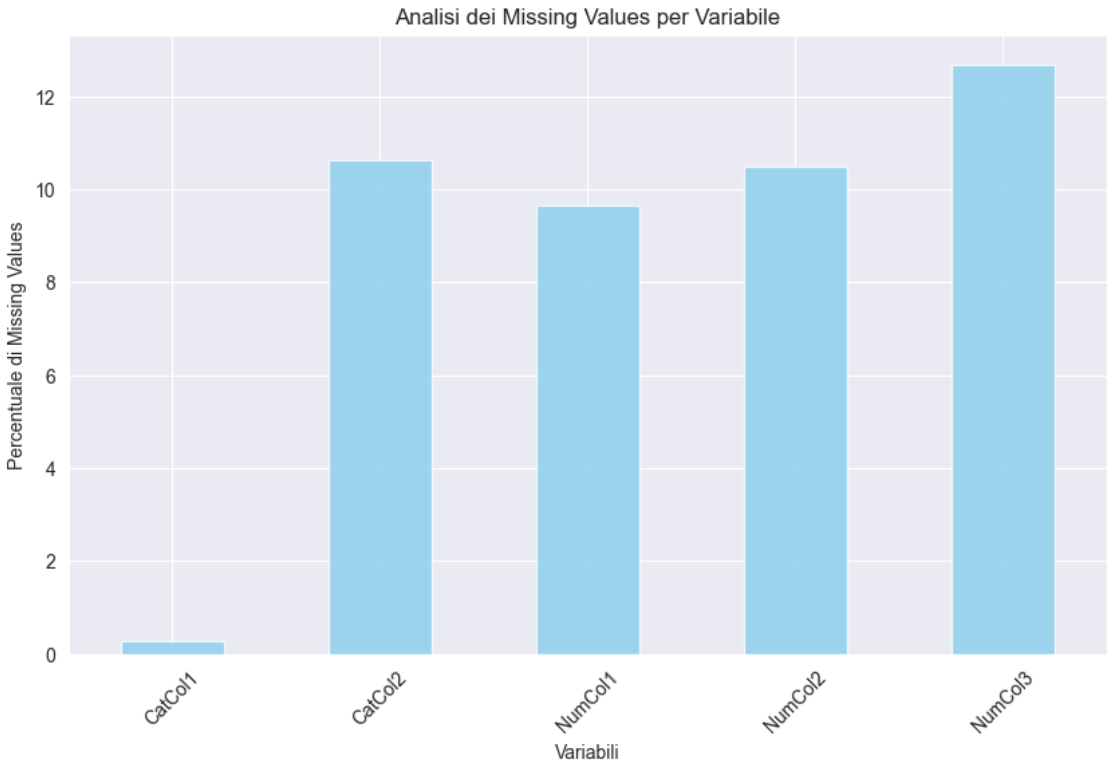
[31]:
```
missing_percent = (df.isnull().sum() / len(df)) * 100
missing_percent
```

[31]:
```
CatCol1     0.29
CatCol2    10.63
NumCol1     9.67
```

```
NumCol2    10.48
NumCol3    12.69
dtype: float64
```

1.23 Grafico della variabile

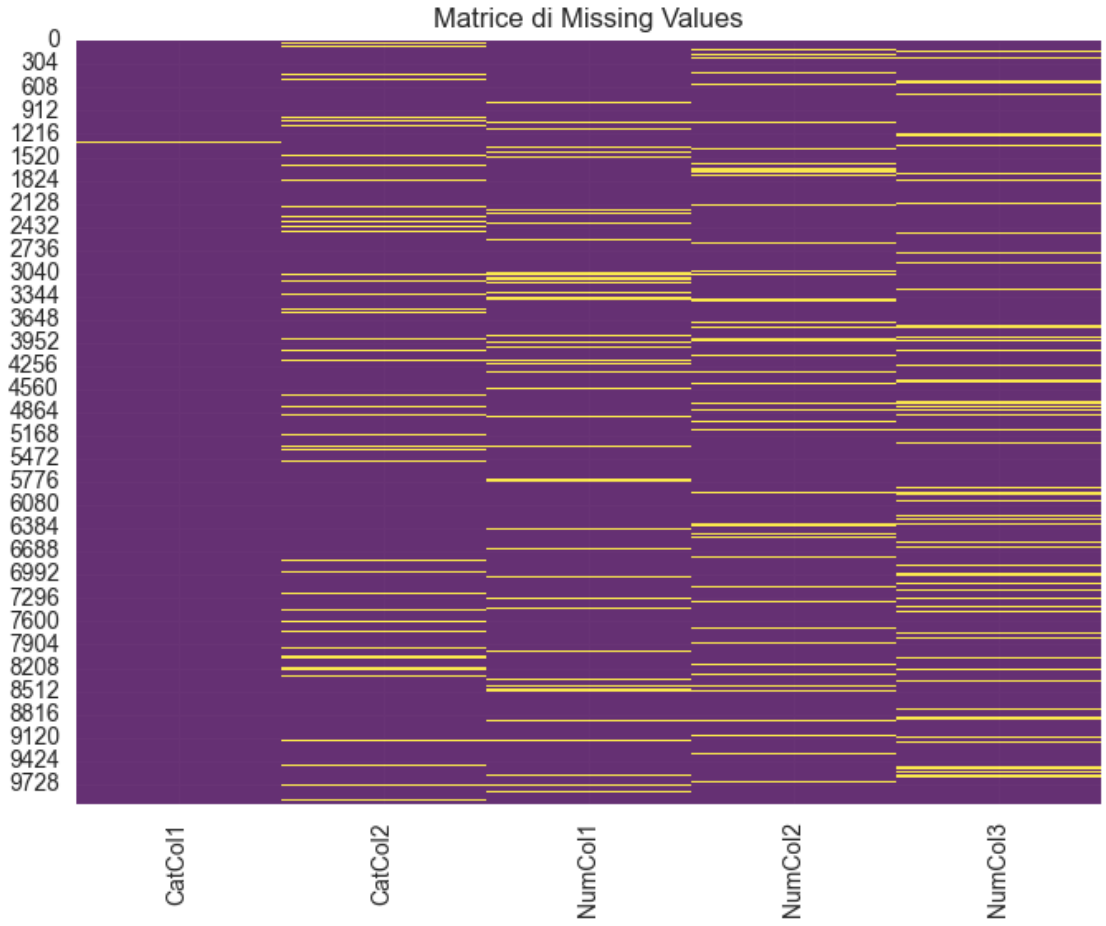
[32]:
```
plt.figure(figsize=(10, 6))
missing_percent.plot(kind='bar', color='skyblue',alpha=0.8)
plt.xlabel('Variabili')
plt.ylabel('Percentuale di Missing Values')
plt.title('Analisi dei Missing Values per Variabile')
plt.xticks(rotation=45)
plt.show()
```



1.24 Grafico Matrice

[33]:
```
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.xticks(rotation=90)

plt.show()
```



1.25 Eliminazione delle righe mancanti in CatCol1 e CatCol2

[34]:
```
df = df.dropna(subset=["CatCol1", 'CatCol2'], how='all')
df
```

[34]:

	CatCol1	CatCol2	NumCol1	NumCol2	NumCol3
0	A	NaN	0.440877	49.0	0.246007
1	A	Y	1.945879	28.0	0.936825
2	C	X	0.988834	42.0	0.751516
3	A	Y	-0.181978	73.0	0.950696
4	B	X	2.080615	74.0	0.903045
...	...	...	...	...	...
9995	C	Y	1.352114	61.0	0.728445

```
9996       C       Y  1.143642     67.0  0.605930
9997       A       X -0.665794     54.0  0.071041
9998       C       Y  0.004278      NaN       NaN
9999       A       X  0.622473     95.0  0.751384

[9995 rows x 5 columns]
```

1.26 Eliminazione delle righe mancanti di NumCol1, NumCol2 e NumCol3

[35]:
```
df = df.dropna(subset=["NumCol1", "NumCol2","NumCol3"], how='all')
df
```

[35]:
```
     CatCol1 CatCol2   NumCol1  NumCol2   NumCol3
0          A     NaN  0.440877     49.0  0.246007
1          A       Y  1.945879     28.0  0.936825
2          C       X  0.988834     42.0  0.751516
3          A       Y -0.181978     73.0  0.950696
4          B       X  2.080615     74.0  0.903045
...      ...     ...       ...      ...       ...
9995       C       Y  1.352114     61.0  0.728445
9996       C       Y  1.143642     67.0  0.605930
9997       A       X -0.665794     54.0  0.071041
9998       C       Y  0.004278      NaN       NaN
9999       A       X  0.622473     95.0  0.751384

[9975 rows x 5 columns]
```

1.27 Trattamento dei valori mancanti: Sostituzione con la moda per le Colonne Categoriche e media condizionata per le Colonne Numeriche

[36]:
```
numeric_cols = df.select_dtypes(include=['number'])
categorical_cols = df.select_dtypes(exclude=['number'])

# Copia esplicità del DataFrame per evitare modifiche indesiderate
df_copy = df.copy()

# Sostituisci i missing values nelle colonne categoriche con la moda
df_copy.loc[:, categorical_cols.columns] = df_copy[categorical_cols.columns].
 ↪fillna(df_copy[categorical_cols.columns].mode().iloc[0])

# Calcola la media condizionata solo per le colonne numeriche con dati mancanti
conditional_means = df_copy.groupby('CatCol1')[numeric_cols.columns].
 ↪transform('mean')

# Aggiorna solo i valori mancanti nelle colonne numeriche con la media␣
 ↪condizionata
```

```
df_copy[numeric_cols.columns] = df_copy[numeric_cols.columns].
 ↪fillna(conditional_means)

# Stampare il DataFrame risultante
print(df_copy)
```

```
     CatCol1 CatCol2   NumCol1    NumCol2   NumCol3
0          A       Y  0.440877  49.000000  0.246007
1          A       Y  1.945879  28.000000  0.936825
2          C       X  0.988834  42.000000  0.751516
3          A       Y -0.181978  73.000000  0.950696
4          B       X  2.080615  74.000000  0.903045
...      ...     ...       ...        ...       ...
9995       C       Y  1.352114  61.000000  0.728445
9996       C       Y  1.143642  67.000000  0.605930
9997       A       X -0.665794  54.000000  0.071041
9998       C       Y  0.004278  49.845018  0.489352
9999       A       X  0.622473  95.000000  0.751384

[9975 rows x 5 columns]
```

1.28 Generatore di dati casuali con sustituzione di moda e mediana

[37]:
```
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 = 10000000
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(0.05 * 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

# Elimina le righe in cui entrambe le features categoriche hanno valori NaN
df = df.dropna(subset=["CatCol1", 'CatCol2'], how='all')
df = df.dropna(subset=["NumCol1", 'NumCol2','NumCol3'], how='all')


numeric_cols = df.select_dtypes(include=['number'])
categorical_cols = df.select_dtypes(exclude=['number'])

# Sostituisci i missing values nelle colonne categoriche con la moda utilizzando␣
 ↪.loc
df.loc[:, categorical_cols.columns] = df[categorical_cols.columns].
 ↪fillna(df[categorical_cols.columns].mode().iloc[0])

# Calcola la media condizionata solo per le colonne numeriche con dati mancanti
conditional_means = df[numeric_cols.columns].fillna(df.
 ↪groupby('CatCol1')[numeric_cols.columns].transform('mean'))

# Aggiorna le colonne numeriche con la media condizionata utilizzando .loc
df.loc[:, numeric_cols.columns] = conditional_means

# Stampa il DataFrame risultante
print(df)
```

```
        CatCol1 CatCol2   NumCol1  NumCol2   NumCol3
0             A       Y -0.391604     98.0  0.409815
1             A       X  0.000551     19.0  0.886592
2             C       Y  1.266001     52.0  0.848556
3             A       X  0.449617     70.0  0.546525
4             B       X  0.742505     72.0  0.467257
...         ...     ...       ...      ...       ...
9999995       A       Y  0.464663      7.0  0.992815
9999996       A       X  0.149775     13.0  0.731368
9999997       C       Y -0.608376      1.0  0.606349
9999998       C       Y  0.000101     69.0  0.115812
9999999       B       Y  1.666715     76.0  0.245699

[9635330 rows x 5 columns]
```

2 Import dati

[38]:
```
import pandas as pd

# Legge il file CSV e lo salva in un dataframe
df1 = pd.read_csv('Pokemon.csv')

# Mostra le prime righe del dataframe per verificare l'importazione
df1.head()
```

[38]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

2.1 Visione generale del dataframe

[39]:
```
print(df1.info())
print(df1.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   #           800 non-null    int64
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      414 non-null    object
 4   Total       800 non-null    int64
 5   HP          800 non-null    int64
 6   Attack      800 non-null    int64
 7   Defense     800 non-null    int64
 8   Sp. Atk     800 non-null    int64
 9   Sp. Def     800 non-null    int64
 10  Speed       800 non-null    int64
 11  Generation  800 non-null    int64
 12  Legendary   800 non-null    bool
dtypes: bool(1), int64(9), object(3)
```

```
memory usage: 75.9+ KB
None
                #      Total          HP      Attack     Defense     Sp. Atk  \
count  800.000000  800.00000  800.000000  800.000000  800.000000  800.000000
mean   362.813750  435.10250   69.258750   79.001250   73.842500   72.820000
std    208.343798  119.96304   25.534669   32.457366   31.183501   32.722294
min      1.000000  180.00000    1.000000    5.000000    5.000000   10.000000
25%    184.750000  330.00000   50.000000   55.000000   50.000000   49.750000
50%    364.500000  450.00000   65.000000   75.000000   70.000000   65.000000
75%    539.250000  515.00000   80.000000  100.000000   90.000000   95.000000
max    721.000000  780.00000  255.000000  190.000000  230.000000  194.000000

          Sp. Def       Speed  Generation
count  800.000000  800.000000   800.00000
mean    71.902500   68.277500     3.32375
std     27.828916   29.060474     1.66129
min     20.000000    5.000000     1.00000
25%     50.000000   45.000000     2.00000
50%     70.000000   65.000000     3.00000
75%     90.000000   90.000000     5.00000
max    230.000000  180.000000     6.00000
```

2.2 Filtraggio dei dati

[40]:
```
# Seleziona una singola colonna
print(df1['Name'])

# Seleziona righe basate su condizioni
print(df1[df1['HP'] > 150])
```

```
0                 Bulbasaur
1                   Ivysaur
2                  Venusaur
3      VenusaurMega Venusaur
4                Charmander
              ...
795                 Diancie
796      DiancieMega Diancie
797      HoopaHoopa Confined
798       HoopaHoopa Unbound
799                Volcanion
Name: Name, Length: 800, dtype: object
       #       Name   Type 1 Type 2  Total   HP  Attack  Defense  Sp. Atk  \
121  113    Chansey   Normal    NaN    450  250       5        5       35
155  143    Snorlax   Normal    NaN    540  160     110       65       65
217  202  Wobbuffet  Psychic    NaN    405  190      33       58       33
261  242    Blissey   Normal    NaN    540  255      10       10       75
351  321    Wailord    Water    NaN    500  170      90       45       90
```

```
655  594  Alomomola    Water    NaN    470  165      75       80       40

     Sp. Def  Speed  Generation  Legendary
121      105     50           1      False
155      110     30           1      False
217       58     33           2      False
261      135     55           2      False
351       45     60           3      False
655       45     65           5      False
```

[41]:
```
import pandas as pd

df = pd.read_csv('Serie A.csv')

df.head()
```

[41]:
```
  Div        Date HomeTeam  AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  ...  \
0  I1  18/08/2018   Chievo  Juventus     2     3   A     1     1   D  ...
1  I1  18/08/2018    Lazio    Napoli     1     2   A     1     1   D  ...
2  I1  19/08/2018  Bologna      Spal     0     1   A     0     0   D  ...
3  I1  19/08/2018   Empoli  Cagliari     2     0   H     1     0   H  ...
4  I1  19/08/2018    Parma   Udinese     2     2   D     1     0   H  ...

   BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA   PSCH  PSCD  \
0      2.13    19   2.00     1.68     1.64     2.38     2.29  18.84  6.42
1      2.17    20   0.00     2.12     2.07     1.83     1.79   2.78  3.57
2      1.58    19  -0.25     1.97     1.92     1.99     1.94   2.31  3.18
3      1.71    19  -0.25     1.98     1.91     1.98     1.94   2.54  3.42
4      1.65    20   0.00     1.81     1.77     2.18     2.10   2.80  3.24

   PSCA
0  1.22
1  2.59
2  3.59
3  2.95
4  2.78

[5 rows x 61 columns]
```

[42]:
```
import csv

# Apre il file CSV e lo legge
with open('Serie A.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

```
 Div        Date HomeTeam  AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  ...  \
```

```
0  I1  18/08/2018   Chievo  Juventus     2     3   A     1     1   D  ...
1  I1  18/08/2018    Lazio    Napoli     1     2   A     1     1   D  ...
2  I1  19/08/2018  Bologna      Spal     0     1   A     0     0   D  ...
3  I1  19/08/2018   Empoli  Cagliari     2     0   H     1     0   H  ...
4  I1  19/08/2018    Parma   Udinese     2     2   D     1     0   H  ...

 BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA   PSCH  PSCD  \
0      2.13    19   2.00     1.68     1.64     2.38     2.29  18.84  6.42
1      2.17    20   0.00     2.12     2.07     1.83     1.79   2.78  3.57
2      1.58    19  -0.25     1.97     1.92     1.99     1.94   2.31  3.18
3      1.71    19  -0.25     1.98     1.91     1.98     1.94   2.54  3.42
4      1.65    20   0.00     1.81     1.77     2.18     2.10   2.80  3.24

   PSCA
0  1.22
1  2.59
2  3.59
3  2.95
4  2.78
```

[43]:
```python
import numpy as np

# Carica il file CSV in un array con numpy
data = np.genfromtxt('Pokemon.csv', delimiter=',')

# Stampiamo i primi 15 elementi per verificare l'importazione
print(data[:20])
```

```
[[  1.  nan  nan  nan 318.  45.  49.  49.  65.  65.  45.   1.  nan]
 [  2.  nan  nan  nan 405.  60.  62.  63.  80.  80.  60.   1.  nan]
 [  3.  nan  nan  nan 525.  80.  82.  83. 100. 100.  80.   1.  nan]
 [  3.  nan  nan  nan 625.  80. 100. 123. 122. 120.  80.   1.  nan]
 [  4.  nan  nan  nan 309.  39.  52.  43.  60.  50.  65.   1.  nan]
 [  5.  nan  nan  nan 405.  58.  64.  58.  80.  65.  80.   1.  nan]
 [  6.  nan  nan  nan 534.  78.  84.  78. 109.  85. 100.   1.  nan]
 [  6.  nan  nan  nan 634.  78. 130. 111. 130.  85. 100.   1.  nan]
 [  6.  nan  nan  nan 634.  78. 104.  78. 159. 115. 100.   1.  nan]
 [  7.  nan  nan  nan 314.  44.  48.  65.  50.  64.  43.   1.  nan]
 [  8.  nan  nan  nan 405.  59.  63.  80.  65.  80.  58.   1.  nan]
 [  9.  nan  nan  nan 530.  79.  83. 100.  85. 105.  78.   1.  nan]
 [  9.  nan  nan  nan 630.  79. 103. 120. 135. 115.  78.   1.  nan]
 [ 10.  nan  nan  nan 195.  45.  30.  35.  20.  20.  45.   1.  nan]
 [ 11.  nan  nan  nan 205.  50.  20.  55.  25.  25.  30.   1.  nan]
 [ 12.  nan  nan  nan 395.  60.  45.  50.  90.  80.  70.   1.  nan]
 [ 13.  nan  nan  nan 195.  40.  35.  30.  20.  20.  50.   1.  nan]
 [ 14.  nan  nan  nan 205.  45.  25.  50.  25.  25.  35.   1.  nan]
 [ 15.  nan  nan  nan 395.  65.  90.  40.  45.  80.  75.   1.  nan]
```

```
[ 15.  nan  nan  nan 495.  65. 150.  40.  15.  80. 145.   1.  nan]]
```