dataset kaggle

April 2, 2024

1 LAVORO DATAFRAME DA KAGGLE

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

1.1.1 Lettura di un file CSV e creazione di un DataFrame utilizzando pandas

[54]:
```
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\\fuel.
 ↪csv"
df = pd.read_csv(percorso_file_csv)
print(df)
```

```
       vehicle_id  year        make               model  \
0           26587  1984  Alfa Romeo           GT V6 2.5
1           27705  1984  Alfa Romeo           GT V6 2.5
2           26561  1984  Alfa Romeo  Spider Veloce 2000
3           27681  1984  Alfa Romeo  Spider Veloce 2000
4           27550  1984  AM General   DJ Po Vehicle 2WD
…             …   …          …                  …
38108       37564  2017       Volvo            XC60 FWD
38109       37547  2017       Volvo            XC90 AWD
38110       37548  2017       Volvo            XC90 AWD
38111       37703  2017       Volvo       XC90 AWD PHEV
38112       37687  2017       Volvo            XC90 FWD

                                    class              drive  \
0                         Minicompact Cars                NaN
1                         Minicompact Cars                NaN
2                              Two Seaters                NaN
3                              Two Seaters                NaN
4              Special Purpose Vehicle 2WD      2-Wheel Drive
…                                        …                …
38108      Small Sport Utility Vehicle 2WD  Front-Wheel Drive
38109    Standard Sport Utility Vehicle 4WD    All-Wheel Drive
38110    Standard Sport Utility Vehicle 4WD    All-Wheel Drive
```

```
38111  Standard Sport Utility Vehicle 4WD    All-Wheel Drive
38112  Standard Sport Utility Vehicle 2WD  Front-Wheel Drive

            transmission transmission_type  engine_index engine_descriptor  \
0         Manual 5-Speed               NaN          9001             (FFS)
1         Manual 5-Speed               NaN          9005    (FFS) CA model
2         Manual 5-Speed               NaN          9002             (FFS)
3         Manual 5-Speed               NaN          9006    (FFS) CA model
4      Automatic 3-Speed               NaN          1830             (FFS)
…                    …                …           …                …
38108     Automatic (S8)               NaN            90              SIDI
38109     Automatic (S8)               NaN            52              SIDI
38110     Automatic (S8)               NaN            53              SIDI
38111     Automatic (S8)               NaN            54        SIDI; PHEV
38112     Automatic (S8)               NaN            50              SIDI

       …  hours_to_charge_ac_240v  composite_city_mpg composite_highway_mpg  \
0      …                     0.0                   0                    0
1      …                     0.0                   0                    0
2      …                     0.0                   0                    0
3      …                     0.0                   0                    0
4      …                     0.0                   0                    0
…    …                       …                   …                    …
38108  …                     0.0                   0                    0
38109  …                     0.0                   0                    0
38110  …                     0.0                   0                    0
38111  …                     0.0                  29                   32
38112  …                     0.0                   0                    0

       composite_combined_mpg range_ft1 city_range_ft1  highway_range_ft1  \
0                           0         0            0.0                0.0
1                           0         0            0.0                0.0
2                           0         0            0.0                0.0
3                           0         0            0.0                0.0
4                           0         0            0.0                0.0
…                         …       …            …                  …
38108                       0         0            0.0                0.0
38109                       0         0            0.0                0.0
38110                       0         0            0.0                0.0
38111                      30         0            0.0                0.0
38112                       0         0            0.0                0.0

       range_ft2  city_range_ft2  highway_range_ft2
0            NaN            0.00                0.0
1            NaN            0.00                0.0
2            NaN            0.00                0.0
3            NaN            0.00                0.0
4            NaN            0.00                0.0
```

```
…          …              …                  …
38108        NaN           0.00                0.0
38109        NaN           0.00                0.0
38110        NaN           0.00                0.0
38111        NaN          13.84               13.3
38112        NaN           0.00                0.0

[38113 rows x 81 columns]

C:\Users\joele\AppData\Local\Temp\ipykernel_13440\1323079333.py:6: DtypeWarning:
Columns (7,44) have mixed types. Specify dtype option on import or set
low_memory=False.
  df = pd.read_csv(percorso_file_csv)
```

Il codice legge un file CSV dal percorso specificato e crea un **DataFrame** utilizzando la libreria **pandas**. Ecco una breve spiegazione delle azioni svolte dal codice:

1. **`import pandas as pd`**: Importa la libreria pandas e la assegna all'alias "pd".
2. **`import numpy as np`**: Importa la libreria numpy e la assegna all'alias "np".
3. **`import matplotlib.pyplot as plt`**: Importa la libreria matplotlib e la assegna all'alias "plt".
4. **`import seaborn as sns`**: Importa la libreria seaborn e la assegna all'alias "sns".
5. **`percorso_file_csv = "C:\\Users\\joele\\OneDrive\\Desktop\\dati robotica\\fuel.csv"`**: Specifica il percorso del file CSV che verrà letto.
6. **`df = pd.read_csv(percorso_file_csv)`**: Legge il file CSV dal percorso specificato e crea un DataFrame chiamato "df".
7. **`print(df)`**: Stampa il contenuto del DataFrame "df".

1.1.2 Conteggio delle Occorrenze di Ciascuna Marca nel DataFrame

[2]:
```
import pandas as pd
# Conta quante volte compare ogni marca
conteggio_marche = df['make'].value_counts().idxmax()

print(conteggio_marche)
```

```
Chevrolet
```

Il codice esegue le seguenti operazioni:

1. Importa la libreria **pandas** con l'alias "pd".
2. Conta quante volte compare ogni marca nel DataFrame `df` utilizzando il metodo `value_counts()`.
3. Estrae la marca più frequente utilizzando il metodo `idxmax()`.

1.1.3 Conteggio delle Occorrenze di Ciascuna Marca nel DataFrame

[3]:
```
import pandas as pd
# Conta quante volte compare ogni marca
conteggio_marche = df['make'].value_counts()
```

```
print(conteggio_marche)
```

```
Chevrolet                 3810
Ford                      3155
Dodge                     2531
GMC                       2398
Toyota                    1937
                          …
London Taxi                  1
Panoz Auto-Development       1
Lambda Control Systems       1
E. P. Dutton, Inc.           1
Excalibur Autos              1
Name: make, Length: 133, dtype: int64
```

Il codice esegue le seguenti operazioni:

1. Importa la libreria **pandas** con l'alias "pd".
2. Conta quante volte compare ogni marca nel DataFrame `df` utilizzando il metodo `value_counts()`.
3. Restituisce una **Serie** contenente il numero di occorrenze per ciascuna marca.

1.1.4 Conteggio delle Occorrenze di Ciascun Anno di Produzione nel DataFrame

[4]:
```
import pandas as pd
anno_produzione = df['year'].value_counts().idxmax()

print(anno_produzione)
```

```
1984
```

Il codice esegue le seguenti operazioni:

1. Importa la libreria **pandas** con l'alias "pd".
2. Conta quante volte compare ciascun anno di produzione nel DataFrame `df` utilizzando il metodo `value_counts()`.
3. Estrae l'anno di produzione più frequente utilizzando il metodo `idxmax()`.

1.1.5 Identificazione e Conteggio delle Occorrenze di Ciascun Anno di Produzione nel DataFrame

[5]:
```
import pandas as pd
anno_produzione = df['year'].value_counts()

print(anno_produzione)
```

```
1984    1964
1985    1701
2015    1270
2016    1250
```

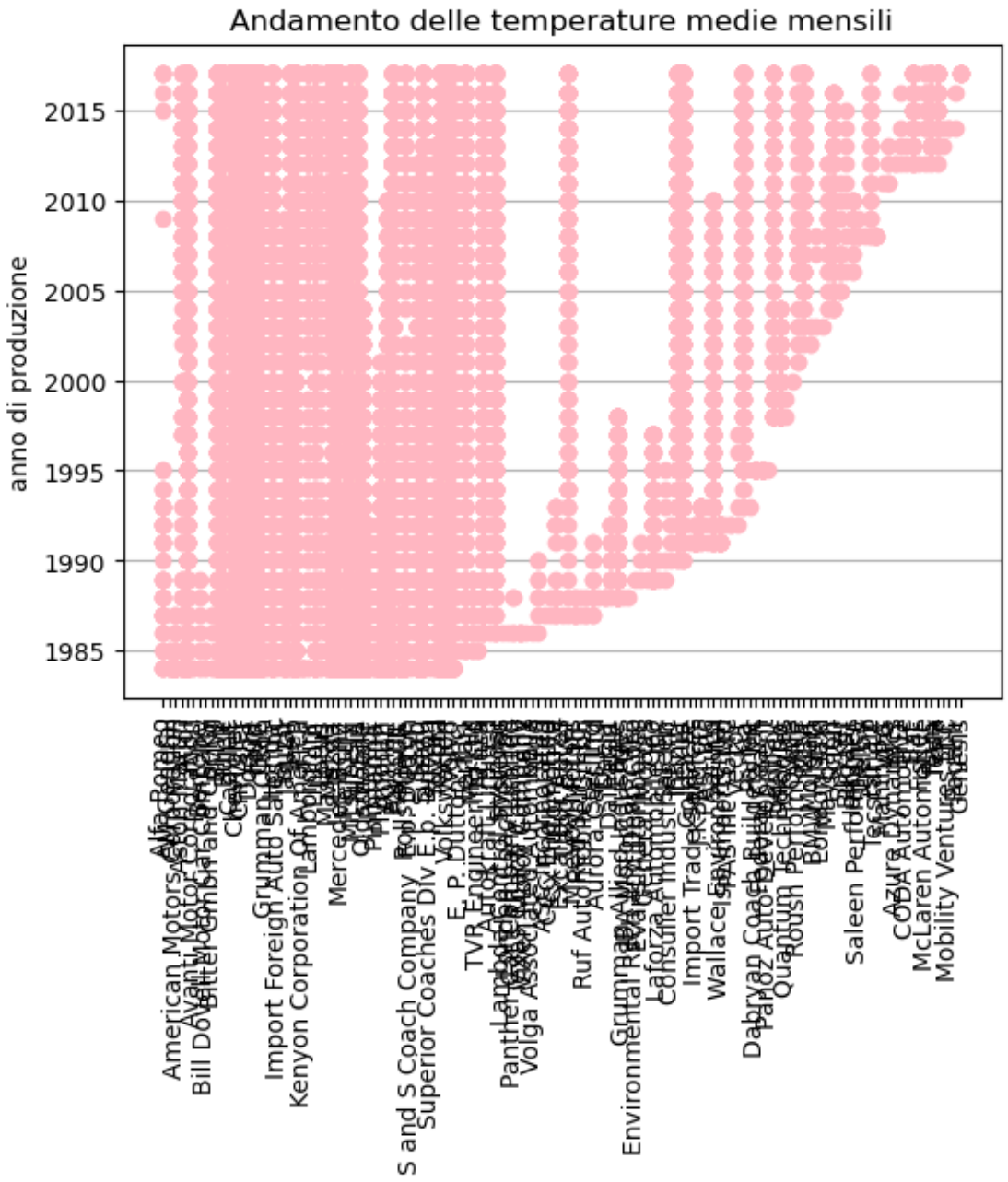
```
1987    1247
2017    1228
1986    1210
2014    1203
2008    1187
2009    1178
2013    1169
2005    1166
1989    1153
2012    1142
1991    1132
1988    1130
2007    1126
2004    1122
2011    1121
1992    1121
2006    1104
2010    1100
1993    1093
1990    1078
2003    1044
1994     982
2002     975
1995     967
2001     911
1999     852
2000     840
1998     812
1996     773
1997     762
Name: year, dtype: int64
```

Il codice esegue le seguenti operazioni:

1. Importa la libreria **pandas** con l'alias "pd".
2. Conta quante volte compare ciascun anno di produzione nel DataFrame `df` utilizzando il metodo `value_counts()`.
3. Restituisce una **Serie** contenente il numero di occorrenze per ciascun anno di produzione.

1.1.6 Creazione e Visualizzazione di un Grafico a Dispersione dei Dati del DataFrame

[6]:
```
import matplotlib.pyplot as plt
plt.plot(df['make'],df['year'], marker='o', linestyle='', color='lightpink')
plt.title('Andamento delle temperature medie mensili')
plt.ylabel('anno di produzione')
plt.xticks(rotation=90)
plt.grid(True, axis="y")
plt.show()
```



Il codice fa quanto segue:

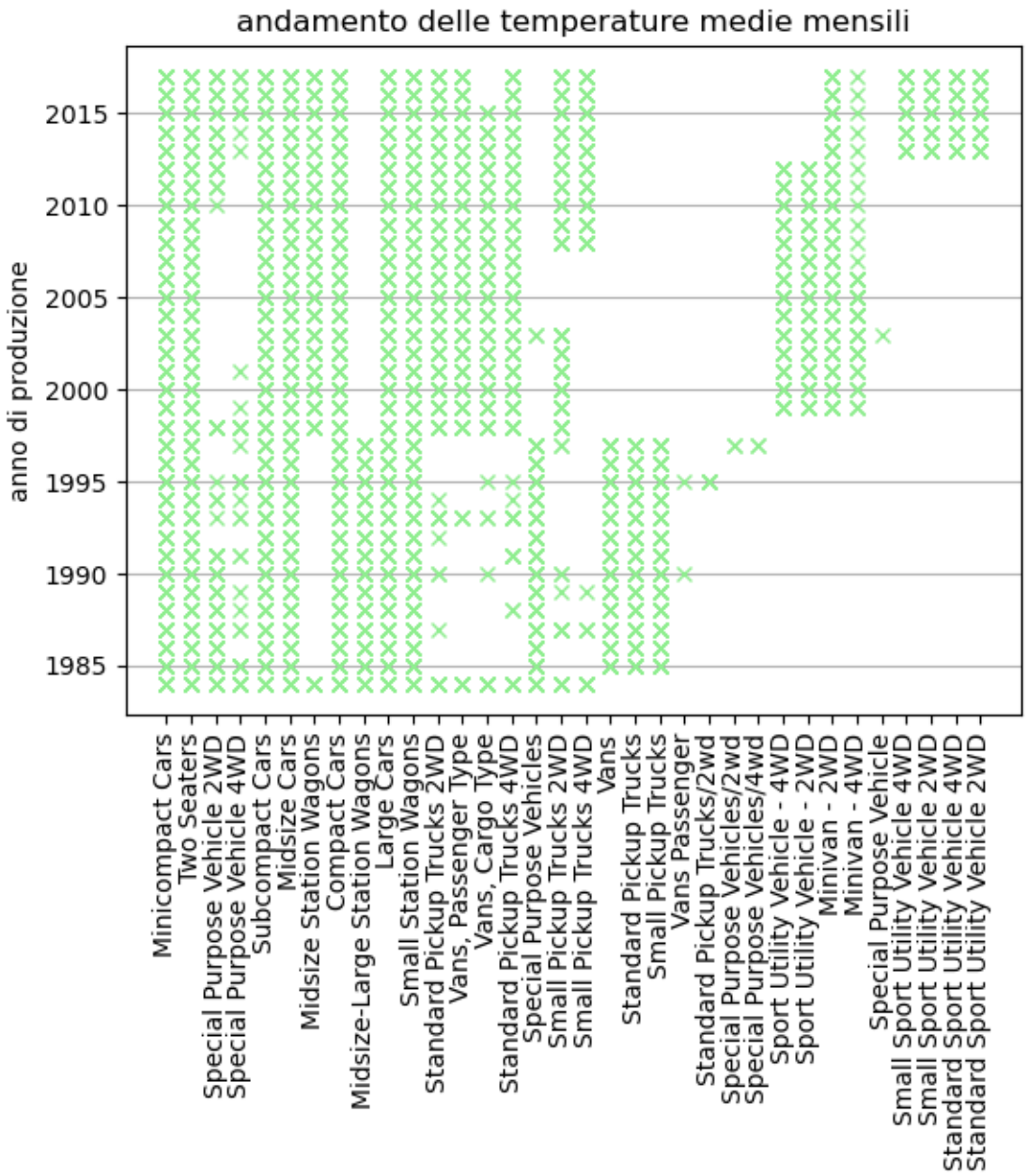
1. **Importa la libreria matplotlib.pyplot**: Questa libreria è utilizzata per creare grafici in Python.
2. **Crea un grafico a dispersione**: Utilizza il metodo `plot` per creare un grafico a dispersione con i dati delle colonne 'make' e 'year' del DataFrame `df`. I punti nel grafico sono rappresentati da cerchi ('o') senza linee di collegamento (' ') e di colore rosa chiaro ('lightpink').
3. **Imposta il titolo del grafico**: Il titolo del grafico è impostato su 'Andamento delle tem-

perature medie mensili' con il metodo `title`.

4. **Imposta l'etichetta dell'asse y**: L'etichetta dell'asse y è impostata su 'anno di produzione' con il metodo `ylabel`.
5. **Ruota le etichette dell'asse x**: Le etichette dell'asse x sono ruotate di 90 gradi per migliorare la leggibilità con il metodo `xticks`.
6. **Aggiunge una griglia all'asse y**: Aggiunge una griglia all'asse y per facilitare la lettura del grafico con il metodo `grid`.
7. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.

1.1.7 Creazione e Visualizzazione di un Grafico a Dispersione dei Dati del DataFrame

[7]:
```
import matplotlib.pyplot as plt

plt.plot(df['class'],df['year'], marker='x', linestyle='', color='lightgreen')
plt.title('andamento delle temperature medie mensili')
plt.xlabel('')
plt.ylabel('anno di produzione')
plt.xticks(rotation=90)
plt.grid(True, axis="y")
plt.show()
```



Il codice fa quanto segue:

1. **Importa la libreria matplotlib.pyplot**: Questa libreria è utilizzata per creare grafici in Python.
2. **Crea un grafico a dispersione**: Utilizza il metodo `plot` per creare un grafico a dispersione con i dati delle colonne 'class' e 'year' del DataFrame `df`. I punti nel grafico sono rappresentati da croci ('x') senza linee di collegamento (' ') e di colore verde chiaro ('lightgreen').
3. **Imposta il titolo del grafico**: Il titolo del grafico è impostato su 'andamento delle temperature medie mensili' con il metodo `title`.

4. **Imposta l'etichetta dell'asse y**: L'etichetta dell'asse y è impostata su 'anno di produzione' con il metodo `ylabel`.
5. **Ruota le etichette dell'asse x**: Le etichette dell'asse x sono ruotate di 90 gradi per migliorare la leggibilità con il metodo `xticks`.
6. **Aggiunge una griglia all'asse y**: Aggiunge una griglia all'asse y per facilitare la lettura del grafico con il metodo `grid`.
7. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.

Quindi, il codice crea e visualizza un grafico a dispersione dei dati nelle colonne 'class' e 'year' del DataFrame `df`, con un titolo specifico, etichette degli assi e una griglia sull'asse y.

1.1.8 Identificazione delle Righe con Dati Mancanti nel DataFrame

[8]:
```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

[8]:
```
       vehicle_id  year        make               model  \
0           26587  1984  Alfa Romeo           GT V6 2.5
1           27705  1984  Alfa Romeo           GT V6 2.5
2           26561  1984  Alfa Romeo  Spider Veloce 2000
3           27681  1984  Alfa Romeo  Spider Veloce 2000
4           27550  1984  AM General   DJ Po Vehicle 2WD
…             …   …          …                   …
38108       37564  2017       Volvo            XC60 FWD
38109       37547  2017       Volvo            XC90 AWD
38110       37548  2017       Volvo            XC90 AWD
38111       37703  2017       Volvo       XC90 AWD PHEV
38112       37687  2017       Volvo            XC90 FWD

                                    class              drive  \
0                         Minicompact Cars                NaN
1                         Minicompact Cars                NaN
2                              Two Seaters                NaN
3                              Two Seaters                NaN
4              Special Purpose Vehicle 2WD      2-Wheel Drive
…                                      …                …
38108     Small Sport Utility Vehicle 2WD  Front-Wheel Drive
38109  Standard Sport Utility Vehicle 4WD    All-Wheel Drive
38110  Standard Sport Utility Vehicle 4WD    All-Wheel Drive
38111  Standard Sport Utility Vehicle 4WD    All-Wheel Drive
38112  Standard Sport Utility Vehicle 2WD  Front-Wheel Drive

            transmission transmission_type  engine_index engine_descriptor  \
0         Manual 5-Speed               NaN          9001             (FFS)
1         Manual 5-Speed               NaN          9005    (FFS) CA model
2         Manual 5-Speed               NaN          9002             (FFS)
```

```
3         Manual 5-Speed               NaN          9006    (FFS) CA model
4      Automatic 3-Speed               NaN          1830             (FFS)
…                  …                  …            …                …
38108     Automatic (S8)               NaN            90              SIDI
38109     Automatic (S8)               NaN            52              SIDI
38110     Automatic (S8)               NaN            53              SIDI
38111     Automatic (S8)               NaN            54        SIDI; PHEV
38112     Automatic (S8)               NaN            50              SIDI

       …  hours_to_charge_ac_240v  composite_city_mpg composite_highway_mpg  \
0      …                      0.0                   0                     0
1      …                      0.0                   0                     0
2      …                      0.0                   0                     0
3      …                      0.0                   0                     0
4      …                      0.0                   0                     0
…    …                        …                     …                     …
38108  …                      0.0                   0                     0
38109  …                      0.0                   0                     0
38110  …                      0.0                   0                     0
38111  …                      0.0                  29                    32
38112  …                      0.0                   0                     0

       composite_combined_mpg range_ft1 city_range_ft1  highway_range_ft1  \
0                           0         0            0.0                0.0
1                           0         0            0.0                0.0
2                           0         0            0.0                0.0
3                           0         0            0.0                0.0
4                           0         0            0.0                0.0
…                         …       …            …                  …
38108                       0         0            0.0                0.0
38109                       0         0            0.0                0.0
38110                       0         0            0.0                0.0
38111                      30         0            0.0                0.0
38112                       0         0            0.0                0.0

       range_ft2  city_range_ft2  highway_range_ft2
0            NaN            0.00                0.0
1            NaN            0.00                0.0
2            NaN            0.00                0.0
3            NaN            0.00                0.0
4            NaN            0.00                0.0
…          …              …                   …
38108        NaN            0.00                0.0
38109        NaN            0.00                0.0
38110        NaN            0.00                0.0
38111        NaN           13.84               13.3
38112        NaN            0.00                0.0
```

```
[38113 rows x 81 columns]
```

Il codice fa quanto segue:

1. **Identifica le righe con dati mancanti**: 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.
2. **Seleziona le righe con dati mancanti**: 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.

Quindi, codice identifica le righe con dati mancanti nel DataFrame `df` e le seleziona in un nuovo DataFrame `righe_con_dati_mancanti`.

1.1.9 Calcolo del Numero Totale di Righe con Dati Mancanti nel DataFrame

[9]:
```
totale_dati_mancanti = righe_con_dati_mancanti.shape[0]
totale_dati_mancanti
```

[9]:
```
38113
```

Il codice fa quanto segue:

1. **Calcola il numero totale di righe con dati mancanti**: Utilizza il metodo `shape[0]` sul DataFrame `righe_con_dati_mancanti` per ottenere il numero totale di righe, che rappresenta il totale dei dati mancanti. Questo valore viene assegnato alla variabile `totale_dati_mancanti`.

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

1.1.10 Stampa delle Righe con Dati Mancanti e del Numero Totale di Queste nel DataFrame

[10]:
```
print("righe con dati mancanti:")
print(righe_con_dati_mancanti)
print("totale dati mancanti: ", totale_dati_mancanti)
```

```
righe con dati mancanti:
       vehicle_id  year        make               model  \
0           26587  1984  Alfa Romeo           GT V6 2.5
1           27705  1984  Alfa Romeo           GT V6 2.5
2           26561  1984  Alfa Romeo  Spider Veloce 2000
3           27681  1984  Alfa Romeo  Spider Veloce 2000
4           27550  1984  AM General   DJ Po Vehicle 2WD
…             …   …          …                  …
38108       37564  2017       Volvo            XC60 FWD
38109       37547  2017       Volvo            XC90 AWD
38110       37548  2017       Volvo            XC90 AWD
```

```
38111       37703  2017       Volvo       XC90 AWD PHEV
38112       37687  2017       Volvo            XC90 FWD

                                  class              drive  \
0                       Minicompact Cars                NaN
1                       Minicompact Cars                NaN
2                            Two Seaters                NaN
3                            Two Seaters                NaN
4              Special Purpose Vehicle 2WD      2-Wheel Drive
…                                      …                …
38108     Small Sport Utility Vehicle 2WD  Front-Wheel Drive
38109  Standard Sport Utility Vehicle 4WD    All-Wheel Drive
38110  Standard Sport Utility Vehicle 4WD    All-Wheel Drive
38111  Standard Sport Utility Vehicle 4WD    All-Wheel Drive
38112  Standard Sport Utility Vehicle 2WD  Front-Wheel Drive

            transmission transmission_type  engine_index engine_descriptor  \
0          Manual 5-Speed               NaN          9001             (FFS)
1          Manual 5-Speed               NaN          9005    (FFS) CA model
2          Manual 5-Speed               NaN          9002             (FFS)
3          Manual 5-Speed               NaN          9006    (FFS) CA model
4       Automatic 3-Speed               NaN          1830             (FFS)
…                       …                …            …                …
38108      Automatic (S8)               NaN            90              SIDI
38109      Automatic (S8)               NaN            52              SIDI
38110      Automatic (S8)               NaN            53              SIDI
38111      Automatic (S8)               NaN            54       SIDI; PHEV
38112      Automatic (S8)               NaN            50              SIDI

       …  hours_to_charge_ac_240v  composite_city_mpg composite_highway_mpg  \
0      …                      0.0                   0                     0
1      …                      0.0                   0                     0
2      …                      0.0                   0                     0
3      …                      0.0                   0                     0
4      …                      0.0                   0                     0
…    …                        …                   …                     …
38108  …                      0.0                   0                     0
38109  …                      0.0                   0                     0
38110  …                      0.0                   0                     0
38111  …                      0.0                  29                    32
38112  …                      0.0                   0                     0

       composite_combined_mpg range_ft1 city_range_ft1  highway_range_ft1  \
0                           0         0            0.0                0.0
1                           0         0            0.0                0.0
2                           0         0            0.0                0.0
3                           0         0            0.0                0.0
4                           0         0            0.0                0.0
```

```
…                          …       …             …                  …
38108                      0       0           0.0                0.0
38109                      0       0           0.0                0.0
38110                      0       0           0.0                0.0
38111                     30       0           0.0                0.0
38112                      0       0           0.0                0.0

       range_ft2  city_range_ft2  highway_range_ft2
0            NaN            0.00                0.0
1            NaN            0.00                0.0
2            NaN            0.00                0.0
3            NaN            0.00                0.0
4            NaN            0.00                0.0
…            …              …                   …
38108        NaN            0.00                0.0
38109        NaN            0.00                0.0
38110        NaN            0.00                0.0
38111        NaN           13.84               13.3
38112        NaN            0.00                0.0

[38113 rows x 81 columns]
totale dati mancanti:  38113
```

Il codice fa quanto segue:

1. **Stampa le righe con dati mancanti**: Utilizza il comando `print` per stampare il DataFrame `righe_con_dati_mancanti`, che contiene tutte le righe del DataFrame `df` che hanno almeno un valore mancante.
2. **Stampa il totale dei dati mancanti**: Utilizza il comando `print` per stampare il valore della variabile `totale_dati_mancanti`, che rappresenta il numero totale di righe con dati mancanti nel DataFrame `df`.

Quindi, il codice stampa le righe con dati mancanti e il numero totale di queste righe nel DataFrame `df`.

1.1.11 Rimozione delle Righe con Dati Mancanti dal DataFrame

[11]:
```
df1=df.dropna(inplace=False)
df1
```

[11]:
```
Empty DataFrame
Columns: [vehicle_id, year, make, model, class, drive, transmission,
transmission_type, engine_index, engine_descriptor, engine_cylinders,
engine_displacement, turbocharger, supercharger, fuel_type, fuel_type_1,
fuel_type_2, city_mpg_ft1, unrounded_city_mpg_ft1, city_mpg_ft2,
unrounded_city_mpg_ft2, city_gasoline_consumption_cd,
city_electricity_consumption, city_utility_factor, highway_mpg_ft1,
unrounded_highway_mpg_ft1, highway_mpg_ft2, unrounded_highway_mpg_ft2,
```

```
highway_gasoline_consumption_cd, highway_electricity_consumption,
highway_utility_factor, unadjusted_city_mpg_ft1, unadjusted_highway_mpg_ft1,
unadjusted_city_mpg_ft2, unadjusted_highway_mpg_ft2, combined_mpg_ft1,
unrounded_combined_mpg_ft1, combined_mpg_ft2, unrounded_combined_mpg_ft2,
combined_electricity_consumption, combined_gasoline_consumption_cd,
combined_utility_factor, annual_fuel_cost_ft1, annual_fuel_cost_ft2,
gas_guzzler_tax, save_or_spend_5_year, annual_consumption_in_barrels_ft1,
annual_consumption_in_barrels_ft2, tailpipe_co2_ft1,
tailpipe_co2_in_grams_mile_ft1, tailpipe_co2_ft2,
tailpipe_co2_in_grams_mile_ft2, fuel_economy_score, ghg_score,
ghg_score_alt_fuel, my_mpg_data, x2d_passenger_volume, x2d_luggage_volume,
x4d_passenger_volume, x4d_luggage_volume, hatchback_passenger_volume,
hatchback_luggage_volume, start_stop_technology, alternative_fuel_technology,
electric_motor, manufacturer_code, gasoline_electricity_blended_cd,
vehicle_charger, alternate_charger, hours_to_charge_120v, hours_to_charge_240v,
hours_to_charge_ac_240v, composite_city_mpg, composite_highway_mpg,
composite_combined_mpg, range_ft1, city_range_ft1, highway_range_ft1, range_ft2,
city_range_ft2, highway_range_ft2]
Index: []

[0 rows x 81 columns]
```

Il codice fa quanto segue:

1. **Rimuove le righe con dati mancanti**: Utilizza il metodo `dropna(inplace=False)` sul DataFrame `df` per rimuovere tutte le righe che contengono almeno un valore mancante. Il parametro `inplace=False` significa che questa operazione non modifica il DataFrame `df` originale, ma restituisce un nuovo DataFrame con le righe rimosse. Questo nuovo DataFrame viene assegnato alla variabile `df1`.

Quindi, il codice rimuove tutte le righe con dati mancanti dal DataFrame `df` e assegna il risultato al DataFrame `df1`.

1.1.12 Creazione di una Matrice di Valori Mancanti per il DataFrame

[12]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
missing_matrix = df.isnull()
missing_matrix
```

[12]:

	vehicle_id	year	make	model	class	drive	transmission	\
0	False	False	False	False	False	True	False	
1	False	False	False	False	False	True	False	
2	False	False	False	False	False	True	False	
3	False	False	False	False	False	True	False	
4	False	False	False	False	False	False	False	
…	…	…	…	…	…	…	…	

```
38108      False  False  False  False  False  False         False
38109      False  False  False  False  False  False         False
38110      False  False  False  False  False  False         False
38111      False  False  False  False  False  False         False
38112      False  False  False  False  False  False         False

       transmission_type  engine_index  engine_descriptor  …  \
0                   True         False              False  …
1                   True         False              False  …
2                   True         False              False  …
3                   True         False              False  …
4                   True         False              False  …
…                   …            …                  …  …
38108               True         False              False  …
38109               True         False              False  …
38110               True         False              False  …
38111               True         False              False  …
38112               True         False              False  …

       hours_to_charge_ac_240v  composite_city_mpg  composite_highway_mpg  \
0                        False               False                  False
1                        False               False                  False
2                        False               False                  False
3                        False               False                  False
4                        False               False                  False
…                        …                   …                      …
38108                    False               False                  False
38109                    False               False                  False
38110                    False               False                  False
38111                    False               False                  False
38112                    False               False                  False

       composite_combined_mpg  range_ft1  city_range_ft1  highway_range_ft1  \
0                       False      False           False              False
1                       False      False           False              False
2                       False      False           False              False
3                       False      False           False              False
4                       False      False           False              False
…                       …          …               …                  …
38108                   False      False           False              False
38109                   False      False           False              False
38110                   False      False           False              False
38111                   False      False           False              False
38112                   False      False           False              False

       range_ft2  city_range_ft2  highway_range_ft2
0           True           False              False
```

```
1           True           False              False
2           True           False              False
3           True           False              False
4           True           False              False
…           …              …                  …
38108       True           False              False
38109       True           False              False
38110       True           False              False
38111       True           False              False
38112       True           False              False

[38113 rows x 81 columns]
```

Il codice fa quanto segue:

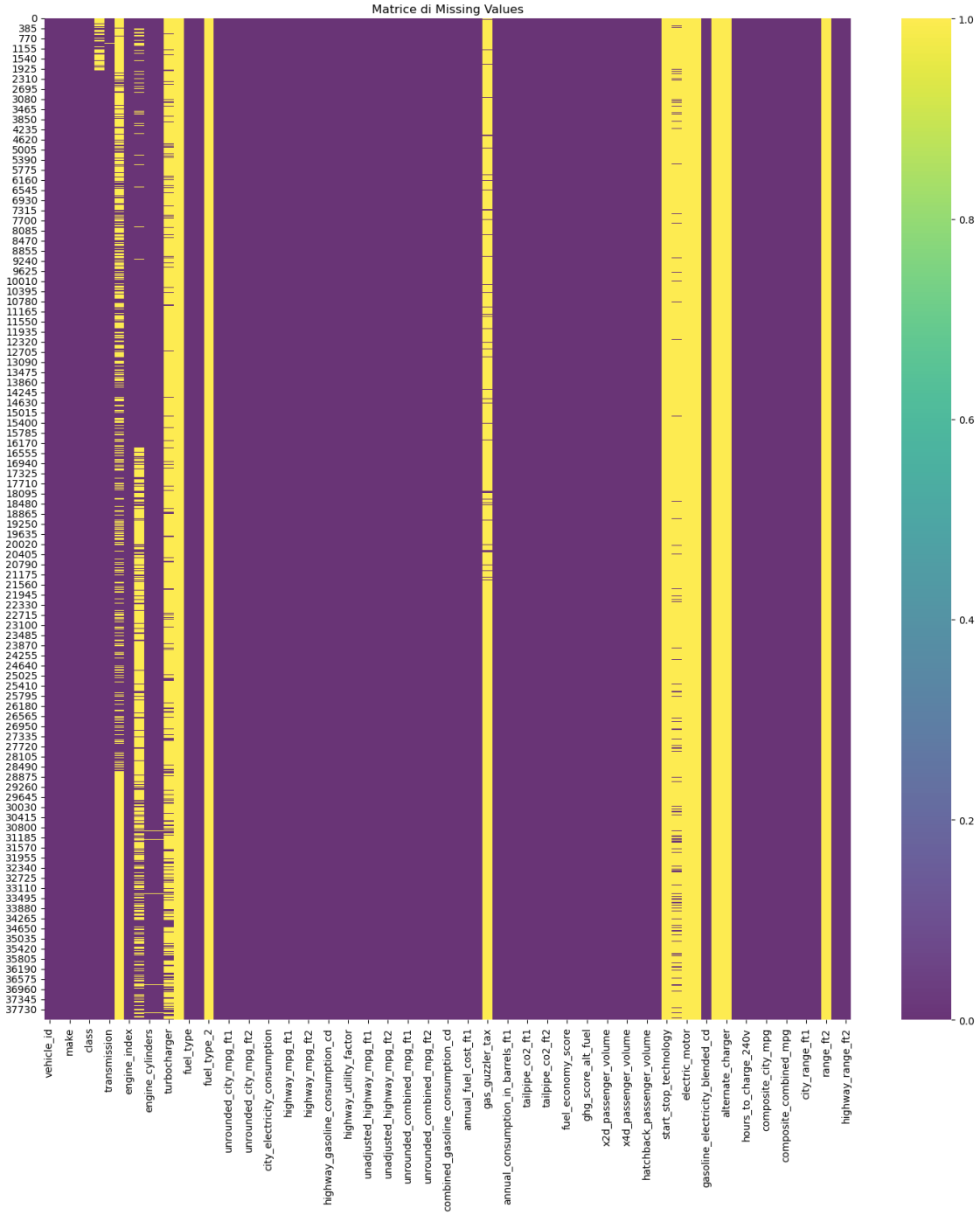
1. **Importa le librerie necessarie**: Importa le librerie pandas, seaborn, matplotlib.pyplot e numpy, che sono utilizzate per la manipolazione dei dati, l'analisi e la visualizzazione dei dati in Python.
2. **Crea una matrice di valori mancanti**: Utilizza il metodo `isnull()` sul DataFrame `df` per creare una matrice booleana (o DataFrame) `missing_matrix` che indica se c'è un valore mancante (NaN) in ciascuna posizione del DataFrame.

Quindi, il codice crea una matrice di valori mancanti per il DataFrame `df`.

1.1.13 Creazione e Visualizzazione di una Heatmap dei Valori Mancanti nel DataFrame

[13]:
```
# Crea una heatmap colorata
plt.figure(figsize=(2^16 ,2^16 ))

sns.heatmap(missing_matrix, cmap='viridis', cbar=True,alpha=0.8)
plt.title('Matrice di Missing Values')
plt.show
```

[13]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Il codice fa quanto segue:

1. **Imposta la dimensione del grafico**: Utilizza il metodo `figure` di matplotlib.pyplot per impostare la dimensione del grafico.

2. **Crea una heatmap**: Utilizza il metodo `heatmap` di seaborn per creare una heatmap (mappa di calore) della matrice `missing_matrix`. La mappa di calore utilizza gradazioni di colore

per rappresentare i valori dei dati nella matrice. Il parametro `cmap='viridis'` imposta la mappa di colori utilizzata per la heatmap. Il parametro `cbar=True` indica che deve essere visualizzata una barra di colore che mostra la corrispondenza tra i colori e i valori dei dati. Il parametro `alpha=0.8` imposta l'opacità dei colori nel grafico.

3. **Imposta il titolo del grafico**: Il titolo del grafico è impostato su 'Matrice di Missing Values' con il metodo `title`.

4. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`. Tuttavia, sembra che ci sia un errore di battitura nel tuo codice. Il comando dovrebbe essere `plt.show()`, non `plt.showù`.

Quindi, il codice crea e visualizza una heatmap dei valori mancanti nel DataFrame `df`, con un titolo specifico.

1.1.14 Selezione delle Colonne Numeriche e Ottenimento dei loro Nomi nel DataFrame

[14]:
```
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols.columns
```

[14]:
```
Index(['vehicle_id', 'year', 'engine_index', 'engine_cylinders',
       'engine_displacement', 'supercharger', 'fuel_type_2', 'city_mpg_ft1',
       'unrounded_city_mpg_ft1', 'city_mpg_ft2', 'unrounded_city_mpg_ft2',
       'city_gasoline_consumption_cd', 'city_electricity_consumption',
       'city_utility_factor', 'highway_mpg_ft1', 'unrounded_highway_mpg_ft1',
       'highway_mpg_ft2', 'unrounded_highway_mpg_ft2',
       'highway_gasoline_consumption_cd', 'highway_electricity_consumption',
       'highway_utility_factor', 'unadjusted_city_mpg_ft1',
       'unadjusted_highway_mpg_ft1', 'unadjusted_city_mpg_ft2',
       'unadjusted_highway_mpg_ft2', 'combined_mpg_ft1',
       'unrounded_combined_mpg_ft1', 'combined_mpg_ft2',
       'unrounded_combined_mpg_ft2', 'combined_electricity_consumption',
       'combined_gasoline_consumption_cd', 'combined_utility_factor',
       'annual_fuel_cost_ft1', 'annual_fuel_cost_ft2', 'save_or_spend_5_year',
       'annual_consumption_in_barrels_ft1',
       'annual_consumption_in_barrels_ft2', 'tailpipe_co2_ft1',
       'tailpipe_co2_in_grams_mile_ft1', 'tailpipe_co2_ft2',
       'tailpipe_co2_in_grams_mile_ft2', 'fuel_economy_score', 'ghg_score',
       'ghg_score_alt_fuel', 'x2d_passenger_volume', 'x2d_luggage_volume',
       'x4d_passenger_volume', 'x4d_luggage_volume',
       'hatchback_passenger_volume', 'hatchback_luggage_volume',
       'start_stop_technology', 'electric_motor', 'manufacturer_code',
       'vehicle_charger', 'alternate_charger', 'hours_to_charge_120v',
       'hours_to_charge_240v', 'hours_to_charge_ac_240v', 'composite_city_mpg',
       'composite_highway_mpg', 'composite_combined_mpg', 'range_ft1',
       'city_range_ft1', 'highway_range_ft1', 'range_ft2', 'city_range_ft2',
       'highway_range_ft2'],
      dtype='object')
```

Il codice fa quanto segue:

1. **Seleziona le colonne numeriche**: Utilizza il metodo `select_dtypes(include=['number'])` sul DataFrame `df` per selezionare tutte le colonne che contengono dati numerici. Questo restituisce un nuovo DataFrame `numeric_cols` che contiene solo le colonne numeriche del DataFrame `df`.
2. **Ottiene i nomi delle colonne**: Utilizza la proprietà `columns` sul DataFrame `numeric_cols` per ottenere una lista con i nomi di tutte le colonne numeriche.

Quindi, il codice seleziona tutte le colonne numeriche nel DataFrame `df` e ottiene i loro nomi.

1.1.15 Calcolo del Numero di Valori Mancanti per Ogni Colonna nel DataFrame

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

[15]:
```
vehicle_id              0
year                    0
make                    0
model                   0
class                   0
                      ...
city_range_ft1          0
highway_range_ft1       0
range_ft2           38113
city_range_ft2          0
highway_range_ft2       0
Length: 81, dtype: int64
```

Il codice fa quanto segue:

1. **Calcola il numero di valori mancanti per ogni colonna**: Utilizza il metodo `isnull().sum()` sul DataFrame `df` per calcolare il numero totale di valori mancanti (NaN) in ogni colonna. Questo restituisce una serie in cui l'indice è il nome della colonna e il valore è il numero di valori mancanti in quella colonna.

Quindi, il codice calcola il numero di valori mancanti in ogni colonna del DataFrame `df`.

1.1.16 Calcolo della Percentuale di Valori Mancanti per Ogni Colonna nel DataFrame

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

[16]:
```
vehicle_id            0.0
year                  0.0
make                  0.0
model                 0.0
class                 0.0
                     ...
city_range_ft1        0.0
```

```
highway_range_ft1      0.0
range_ft2            100.0
city_range_ft2         0.0
highway_range_ft2      0.0
Length: 81, dtype: float64
```

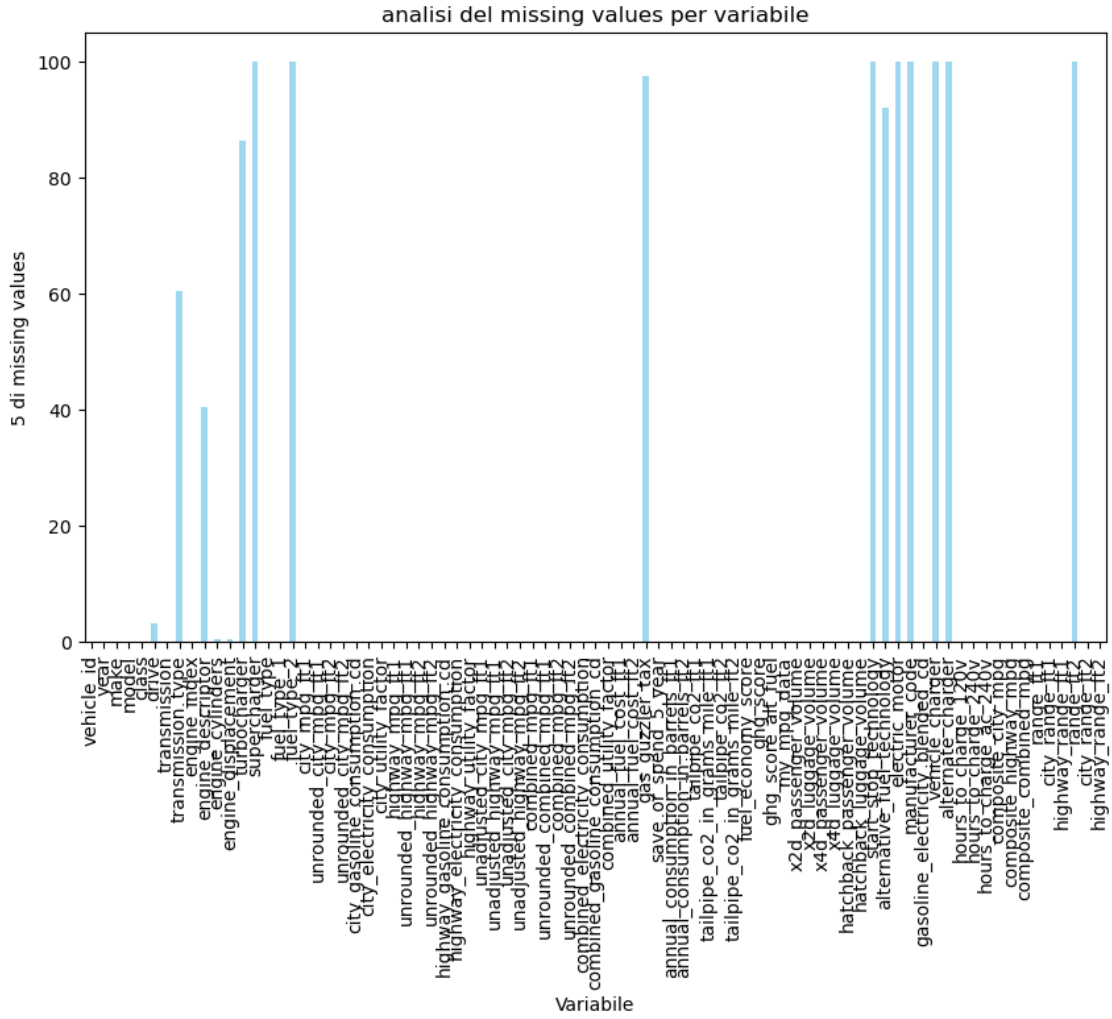
Il codice fa quanto segue:

1. **Calcola la percentuale di valori mancanti per ogni colonna**: Utilizza il metodo `isnull().sum()` sul DataFrame `df` per calcolare il numero totale di valori mancanti (NaN) in ogni colonna. Questo numero viene poi diviso per il numero totale di righe nel DataFrame `df` (ottenuto con `len(df)`) e moltiplicato per 100 per convertirlo in percentuale. Questo restituisce una serie in cui l'indice è il nome della colonna e il valore è la percentuale di valori mancanti in quella colonna. Questa serie viene assegnata alla variabile `missing_percent`.

Quindi, il codice calcola la percentuale di valori mancanti in ogni colonna del DataFrame `df`.

1.1.17 Calcolo della Percentuale di Valori Mancanti per Ogni Variabile e Creazione di un Grafico a Barre

[17]:
```
#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=90)
plt.show()
```



Il codice fa quanto segue:

1. **Calcola la percentuale di righe con valori mancanti per ogni variabile**: Utilizza il metodo `isnull().sum()` sul DataFrame `df` per calcolare il numero totale di valori mancanti (NaN) in ogni colonna. Questo numero viene poi diviso per il numero totale di righe nel DataFrame `df` (ottenuto con `len(df)`) e moltiplicato per 100 per convertirlo in percentuale. Questo restituisce una serie in cui l'indice è il nome della colonna e il valore è la percentuale di valori mancanti in quella colonna. Questa serie viene assegnata alla variabile `missing_percent`.

2. **Crea un grafico a barre**: Utilizza il metodo `plot` con il parametro `kind='bar'` per creare un grafico a barre della serie `missing_percent`. Le barre nel grafico sono di colore azzurro cielo ('skyblue') con un'opacità dell'80% (`alpha=0.8`).

3. **Imposta le etichette degli assi e il titolo del grafico**: Imposta l'etichetta dell'asse x su 'Variabile', l'etichetta dell'asse y su '5 di missing values' e il titolo del grafico su 'analisi del missing values per variabile' con i metodi `xlabel`, `ylabel` e `title`.

4. **Ruota le etichette dell'asse x**: Le etichette dell'asse x sono ruotate di 90 gradi per migliorare la leggibilità con il metodo `xticks`.

5. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.

Quindi, il codice calcola la percentuale di valori mancanti in ogni colonna del DataFrame `df`, crea un grafico a barre di queste percentuali e visualizza il grafico.

1.1.18 Ottenimento di Informazioni sul DataFrame e Calcolo delle Statistiche Descrittive

[18]:
```
#informazioni sul dataset
print(df.info())

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38113 entries, 0 to 38112
Data columns (total 81 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   vehicle_id                        38113 non-null  int64
 1   year                              38113 non-null  int64
 2   make                              38113 non-null  object
 3   model                             38113 non-null  object
 4   class                             38113 non-null  object
 5   drive                             36924 non-null  object
 6   transmission                      38102 non-null  object
 7   transmission_type                 15045 non-null  object
 8   engine_index                      38113 non-null  int64
 9   engine_descriptor                 22693 non-null  object
 10  engine_cylinders                  37977 non-null  float64
 11  engine_displacement               37979 non-null  float64
 12  turbocharger                      5239 non-null   object
 13  supercharger                      0 non-null      float64
 14  fuel_type                         38113 non-null  object
 15  fuel_type_1                       38113 non-null  object
 16  fuel_type_2                       0 non-null      float64
 17  city_mpg_ft1                      38113 non-null  int64
 18  unrounded_city_mpg_ft1            38113 non-null  float64
 19  city_mpg_ft2                      38113 non-null  int64
 20  unrounded_city_mpg_ft2            38113 non-null  float64
 21  city_gasoline_consumption_cd      38113 non-null  float64
 22  city_electricity_consumption      38113 non-null  float64
 23  city_utility_factor               38113 non-null  float64
 24  highway_mpg_ft1                   38113 non-null  int64
 25  unrounded_highway_mpg_ft1         38113 non-null  float64
 26  highway_mpg_ft2                   38113 non-null  int64
```

```
27  unrounded_highway_mpg_ft2          38113 non-null  float64
28  highway_gasoline_consumption_cd    38113 non-null  float64
29  highway_electricity_consumption    38113 non-null  float64
30  highway_utility_factor             38113 non-null  float64
31  unadjusted_city_mpg_ft1            38113 non-null  float64
32  unadjusted_highway_mpg_ft1         38113 non-null  float64
33  unadjusted_city_mpg_ft2            38113 non-null  float64
34  unadjusted_highway_mpg_ft2         38113 non-null  float64
35  combined_mpg_ft1                   38113 non-null  int64
36  unrounded_combined_mpg_ft1         38113 non-null  float64
37  combined_mpg_ft2                   38113 non-null  int64
38  unrounded_combined_mpg_ft2         38113 non-null  float64
39  combined_electricity_consumption   38113 non-null  float64
40  combined_gasoline_consumption_cd   38113 non-null  float64
41  combined_utility_factor            38113 non-null  float64
42  annual_fuel_cost_ft1               38113 non-null  int64
43  annual_fuel_cost_ft2               38113 non-null  int64
44  gas_guzzler_tax                    964 non-null    object
45  save_or_spend_5_year               38113 non-null  int64
46  annual_consumption_in_barrels_ft1  38113 non-null  float64
47  annual_consumption_in_barrels_ft2  38113 non-null  float64
48  tailpipe_co2_ft1                   38113 non-null  int64
49  tailpipe_co2_in_grams_mile_ft1     38113 non-null  float64
50  tailpipe_co2_ft2                   38113 non-null  int64
51  tailpipe_co2_in_grams_mile_ft2     38113 non-null  float64
52  fuel_economy_score                 38113 non-null  int64
53  ghg_score                          38113 non-null  int64
54  ghg_score_alt_fuel                 38113 non-null  int64
55  my_mpg_data                        38113 non-null  object
56  x2d_passenger_volume               38113 non-null  int64
57  x2d_luggage_volume                 38113 non-null  int64
58  x4d_passenger_volume               38113 non-null  int64
59  x4d_luggage_volume                 38113 non-null  int64
60  hatchback_passenger_volume         38113 non-null  int64
61  hatchback_luggage_volume           38113 non-null  int64
62  start_stop_technology              0 non-null      float64
63  alternative_fuel_technology        3047 non-null   object
64  electric_motor                     0 non-null      float64
65  manufacturer_code                  0 non-null      float64
66  gasoline_electricity_blended_cd    38113 non-null  bool
67  vehicle_charger                    0 non-null      float64
68  alternate_charger                  0 non-null      float64
69  hours_to_charge_120v               38113 non-null  int64
70  hours_to_charge_240v               38113 non-null  float64
71  hours_to_charge_ac_240v            38113 non-null  float64
72  composite_city_mpg                 38113 non-null  int64
73  composite_highway_mpg              38113 non-null  int64
74  composite_combined_mpg             38113 non-null  int64
```

```
 75  range_ft1                          38113 non-null  int64
 76  city_range_ft1                     38113 non-null  float64
 77  highway_range_ft1                  38113 non-null  float64
 78  range_ft2                          0 non-null      float64
 79  city_range_ft2                     38113 non-null  float64
 80  highway_range_ft2                  38113 non-null  float64
dtypes: bool(1), float64(39), int64(28), object(13)
memory usage: 23.3+ MB
None
         vehicle_id          year  engine_index  engine_cylinders  \
count  38113.000000  38113.000000  38113.000000      37977.000000
mean   19170.638496   2000.194527   8799.389001          5.736656
std    11134.878665     10.464573  17781.058490          1.752254
min        1.000000   1984.000000      0.000000          2.000000
25%     9529.000000   1991.000000      0.000000          4.000000
50%    19058.000000   2001.000000    212.000000          6.000000
75%    28779.000000   2009.000000   4451.000000          6.000000
max    38542.000000   2017.000000  69102.000000         16.000000

       engine_displacement  supercharger  fuel_type_2  city_mpg_ft1  \
count         37979.000000           0.0          0.0  38113.000000
mean              3.317583           NaN          NaN     17.981109
std               1.361995           NaN          NaN      6.849728
min               0.000000           NaN          NaN      6.000000
25%               2.200000           NaN          NaN     15.000000
50%               3.000000           NaN          NaN     17.000000
75%               4.300000           NaN          NaN     20.000000
max               8.400000           NaN          NaN    150.000000

       unrounded_city_mpg_ft1  city_mpg_ft2  …  hours_to_charge_ac_240v  \
count            38113.000000  38113.000000  …             38113.000000
mean                 4.606426      0.546218  …                 0.005549
std                 10.113963      4.109282  …                 0.161014
min                  0.000000      0.000000  …                 0.000000
25%                  0.000000      0.000000  …                 0.000000
50%                  0.000000      0.000000  …                 0.000000
75%                  0.000000      0.000000  …                 0.000000
max                150.000000    145.000000  …                 7.000000

       composite_city_mpg  composite_highway_mpg  composite_combined_mpg  \
count        38113.000000           38113.000000            38113.000000
mean             0.082203               0.080891                0.081311
std              2.156682               2.052187                2.097794
min              0.000000               0.000000                0.000000
25%              0.000000               0.000000                0.000000
50%              0.000000               0.000000                0.000000
75%              0.000000               0.000000                0.000000
max             97.000000              81.000000               88.000000
```

```
          range_ft1  city_range_ft1  highway_range_ft1  range_ft2  \
count  38113.000000    38113.000000       38113.000000        0.0
mean       0.469708        0.426249           0.419197        NaN
std        9.352069        9.104702           9.315914        NaN
min        0.000000        0.000000           0.000000        NaN
25%        0.000000        0.000000           0.000000        NaN
50%        0.000000        0.000000           0.000000        NaN
75%        0.000000        0.000000           0.000000        NaN
max      315.000000      305.900000         346.900000        NaN

       city_range_ft2  highway_range_ft2
count    38113.000000       38113.000000
mean         0.043973           0.040051
std          1.311628           1.169281
min          0.000000           0.000000
25%          0.000000           0.000000
50%          0.000000           0.000000
75%          0.000000           0.000000
max        103.030000          90.550000

[8 rows x 67 columns]
```

Il codice fa quanto segue:

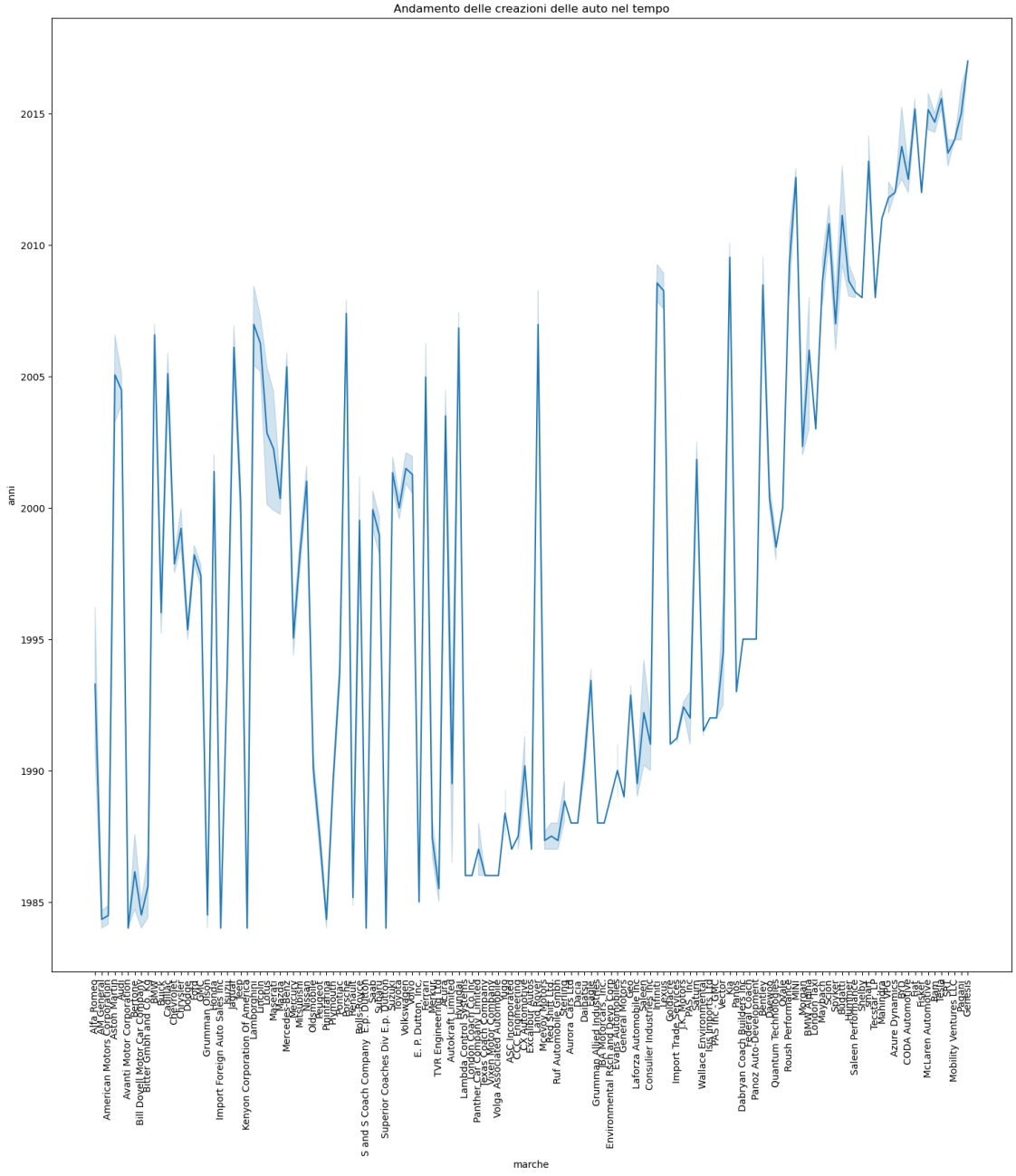
1. **Ottiene informazioni sul DataFrame**: Utilizza il metodo `info()` sul DataFrame `df` per ottenere un riepilogo conciso del DataFrame, compresi il numero totale di elementi non nulli in ogni colonna, il tipo di dati di ogni colonna, la memoria utilizzata dal DataFrame, ecc. Queste informazioni vengono stampate con il comando `print`.
2. **Calcola le statistiche descrittive**: Utilizza il metodo `describe()` sul DataFrame `df` per generare statistiche descrittive che riassumono la tendenza centrale, la dispersione e la forma della distribuzione di un dataset, escludendo i valori `NaN`. Queste statistiche includono il conteggio, la media, lo scarto standard, il minimo, il 25° percentile (Q1), la mediana (50° percentile o Q2), il 75° percentile (Q3) e il massimo. Queste statistiche vengono stampate con il comando `print`.

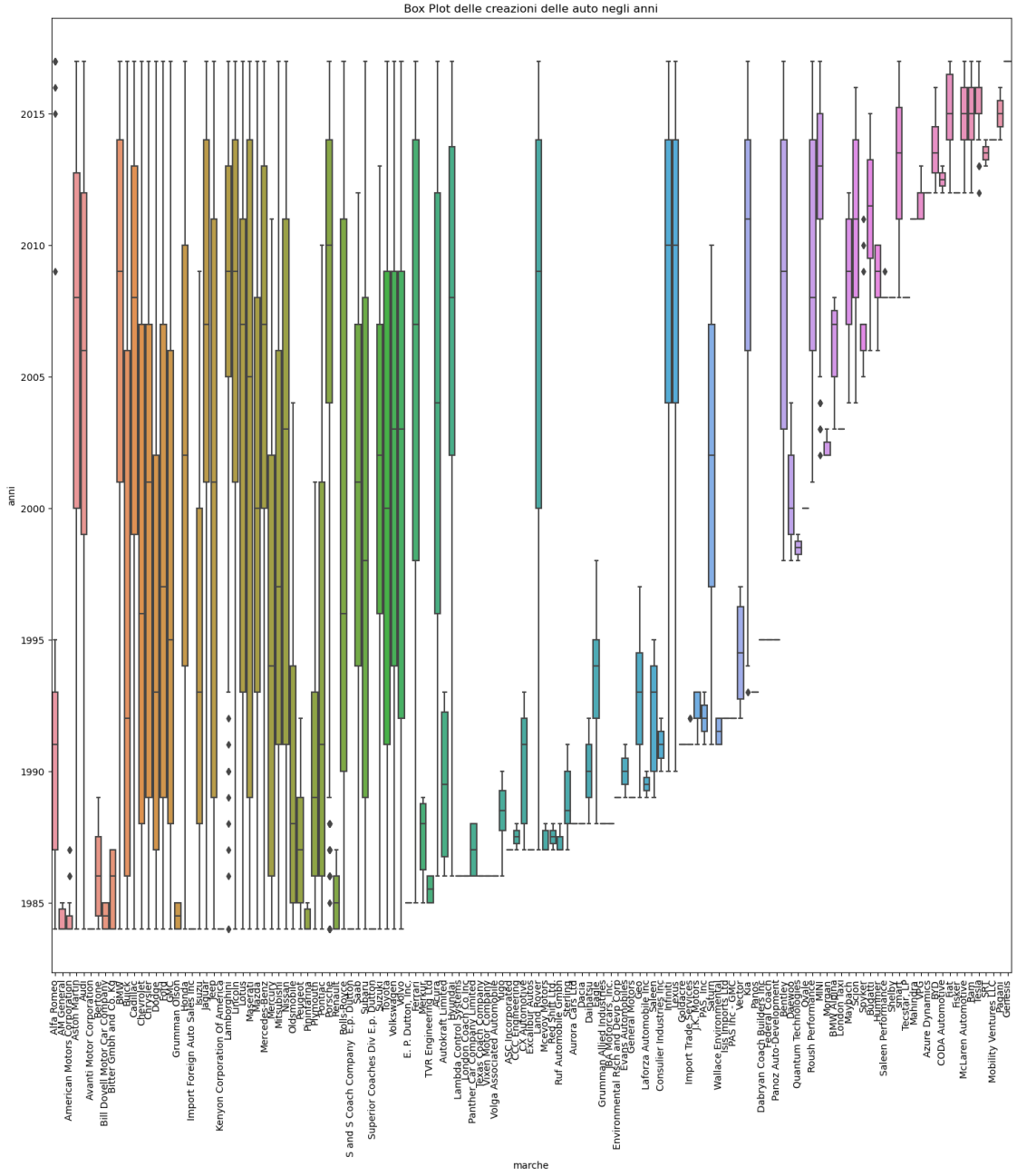
Quindi, il codice fornisce un riepilogo del DataFrame `df` e calcola le statistiche descrittive dei dati nel DataFrame.

1.1.19 Creazione e Visualizzazione di un Grafico a Linee e un Box Plot dei Dati del DataFrame

[19]:
```
# Visualizza un grafico delle vendite nel tempo
plt.figure(figsize=(2^16, 2^16))
sns.lineplot(x='make', y='year', data=df)
plt.title('Andamento delle creazioni delle auto nel tempo')
plt.xlabel('marche')
plt.ylabel('anni')
```

```
plt.xticks(rotation=90)
plt.show()

# Visualizza una box plot delle vendite per prodotto
plt.figure(figsize=(2^16, 2^16))
sns.boxplot(x='make', y='year', data=df)
plt.title('Box Plot delle creazioni delle auto negli anni')
plt.xlabel('marche')
plt.ylabel('anni')
plt.xticks(rotation=90)
plt.show()
```


Andamento delle creazioni delle auto nel tempo
anni
2015
2010
2005
2000
1995
1990
1985
marche



Il codice fa quanto segue:

1. **Imposta la dimensione del grafico**: Utilizza il metodo `figure` di matplotlib.pyplot per impostare la dimensione del grafico. Tuttavia, sembra che ci sia un errore nel codice. In Python, l'operatore `^` non rappresenta l'elevamento a potenza, ma l'operazione bitwise XOR. Per l'elevamento a potenza, dovresti usare l'operatore `**`. Quindi, se intendevi utilizzare 2 elevato alla potenza di 16, il codice corretto sarebbe `plt.figure(figsize=(2**16, 2**16))`.

2. **Crea un grafico a linee**: Utilizza il metodo `lineplot` di seaborn per creare un grafico a

linee con i dati delle colonne 'make' e 'year' del DataFrame `df`.

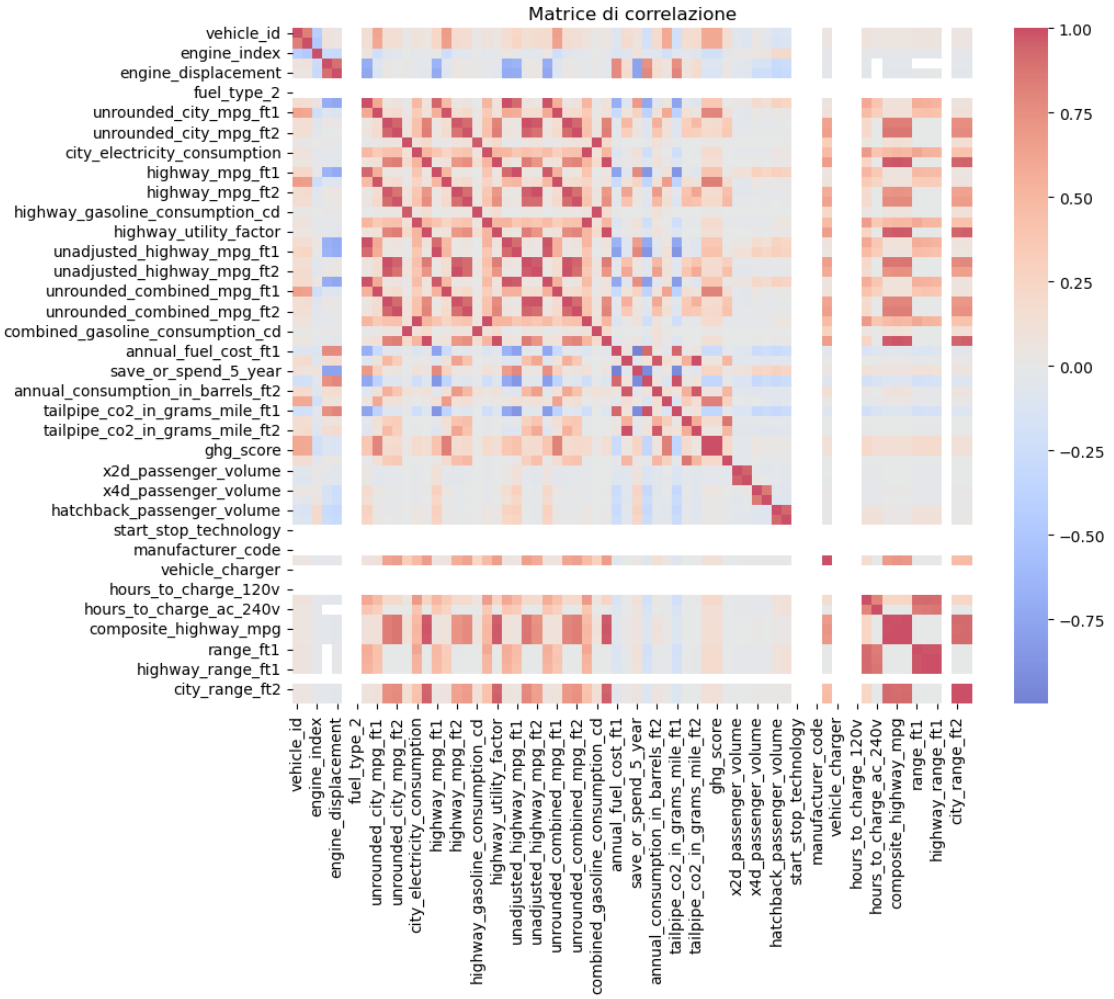
3. **Imposta le etichette degli assi e il titolo del grafico**: Imposta l'etichetta dell'asse x su 'marche', l'etichetta dell'asse y su 'anni' e il titolo del grafico su 'Andamento delle creazioni delle auto nel tempo' con i metodi `xlabel`, `ylabel` e `title`.
4. **Ruota le etichette dell'asse x**: Le etichette dell'asse x sono ruotate di 90 gradi per migliorare la leggibilità con il metodo `xticks`.
5. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.
6. **Crea un box plot**: Utilizza il metodo `boxplot` di seaborn per creare un box plot con i dati delle colonne 'make' e 'year' del DataFrame `df`.
7. **Imposta le etichette degli assi e il titolo del grafico**: Imposta l'etichetta dell'asse x su 'marche', l'etichetta dell'asse y su 'anni' e il titolo del grafico su 'Box Plot delle creazioni delle auto negli anni' con i metodi `xlabel`, `ylabel` e `title`.
8. **Ruota le etichette dell'asse x**: Le etichette dell'asse x sono ruotate di 90 gradi per migliorare la leggibilità con il metodo `xticks`.
9. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.

Quindi, il codice crea e visualizza un grafico a linee e un box plot dei dati nelle colonne 'make' e 'year' del DataFrame `df`, con titoli specifici, etichette degli assi e una rotazione delle etichette dell'asse x.

1.1.20 Calcolo della Matrice di Correlazione e Creazione di una Heatmap

[20]:
```
## import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

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

#visualizza la matrice di correlazione come heatmap
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm', fmt=".2f",␣
 ↪alpha=0.7)

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

```
C:\Users\joele\AppData\Local\Temp\ipykernel_13440\443977044.py:6: 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.
  correlation_matrix = df.corr()
```



Il codice fa quanto segue:

1. **Calcola la matrice di correlazione**: Utilizza il metodo `corr()` sul DataFrame `df` per calcolare la matrice di correlazione tra tutte le variabili numeriche. Questa matrice viene assegnata alla variabile `correlation_matrix`.

2. **Imposta la dimensione del grafico**: Utilizza il metodo `figure` di matplotlib.pyplot per impostare la dimensione del grafico.

3. **Crea una heatmap**: Utilizza il metodo `heatmap` di seaborn per creare una heatmap (mappa di calore) della matrice di correlazione. La mappa di calore utilizza gradazioni di colore per rappresentare i valori dei dati nella matrice. Il parametro `annot=False` indica che i valori dei dati non devono essere visualizzati nel grafico. Il parametro `cmap='coolwarm'` imposta la mappa di colori utilizzata per la heatmap. Il parametro `fmt=".2f"` imposta il formato dei numeri nel grafico. Il parametro `alpha=0.7` imposta l'opacità dei colori nel grafico.

4. **Imposta il titolo del grafico**: Il titolo del grafico è impostato su 'Matrice di correlazione' con il metodo `title`.

5. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.

Quindi, il codice calcola la matrice di correlazione delle variabili numeriche nel DataFrame `df`, crea una heatmap di questa matrice e visualizza il grafico.

1.1.21 Suddivisione del DataFrame in Training Set e Test Set, Creazione di un Grafico a Dispersione e Stampa delle Dimensioni dei Set

[21]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split



# Suddivisione del dataset in training set (70%) e test set (30%)
X_train, X_test, y_train, y_test = train_test_split(df['year'],df['make'],␣
 ↪test_size=0.3, random_state=42)

# Creazione di un grafico a dispersione
plt.figure(figsize=(2^16, 2^16))
plt.scatter(X_train, y_train, label='Training Set', color='blue', alpha=0.7)
plt.scatter(X_test, y_test, label='Test Set', color='orange', alpha=0.7)
plt.xlabel('anno di produzione')
plt.ylabel('marche')
plt.title('Relazione tra Visite al Sito e Importo delle Vendite')
plt.legend()
plt.grid(True)
plt.show()

# Stampare le dimensioni dei training set e test set
print("Dimensioni del Training Set (visite al sito e importo delle vendite):",␣
 ↪X_train.shape, y_train.shape)
print("Dimensioni del Test Set (visite al sito e importo delle vendite):",␣
 ↪X_test.shape, y_test.shape)
```


```
Dimensioni del Training Set (visite al sito e importo delle vendite): (26679,)
(26679,)
Dimensioni del Test Set (visite al sito e importo delle vendite): (11434,)
(11434,)
```

Il codice fa quanto segue:

1. **Suddivisione del dataset in training set e test set**: Utilizza la funzione `train_test_split` della libreria sklearn per suddividere le colonne 'year' e 'make' del DataFrame `df` in un training set (70% dei dati) e un test set (30% dei dati). Questa suddivisione viene fatta in modo casuale, ma la casualità è controllata dal parametro `random_state=42` per garantire che i risultati siano riproducibili.

2. **Crea un grafico a dispersione**: Utilizza il metodo `scatter` di matplotlib.pyplot per creare un grafico a dispersione dei dati del training set e del test set. I punti del training set sono rappresentati in blu e quelli del test set in arancione.

3. **Imposta le etichette degli assi e il titolo del grafico**: Imposta l'etichetta dell'asse x su 'anno di produzione', l'etichetta dell'asse y su 'marche' e il titolo del grafico su 'Relazione tra

Visite al Sito e Importo delle Vendite' con i metodi `xlabel`, `ylabel` e `title`.

4. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.
5. **Stampa le dimensioni dei training set e test set**: Utilizza il comando `print` per stampare le dimensioni del training set e del test set.

Quindi, il codice suddivide il DataFrame `df` in un training set e un test set, crea un grafico a dispersione dei dati e stampa le dimensioni dei set.

1.1.22 Suddivisione del DataFrame in Tre Subset di Dimensioni Simili

[22]:
```
# Creare tre subset di dimensioni simili
subset1 = df.sample(frac=1/3)
df = df.drop(subset1.index)

subset2 = df.sample(frac=1/2)
df = df.drop(subset2.index)

subset3 = df  # L'ultimo subset con il rimanente
```

Il codice fa quanto segue:

1. **Crea il primo subset**: Utilizza il metodo `sample` sul DataFrame `df` con il parametro `frac=1/3` per selezionare un campione casuale di righe che rappresenta un terzo del DataFrame originale. Questo subset viene assegnato alla variabile `subset1`.
2. **Rimuove le righe del primo subset dal DataFrame originale**: Utilizza il metodo `drop` sul DataFrame `df` con l'indice del primo subset come parametro per rimuovere le righe corrispondenti dal DataFrame originale.
3. **Crea il secondo subset**: Ripete i passaggi 1 e 2 per creare un secondo subset, questa volta selezionando la metà delle righe rimanenti nel DataFrame `df`. Questo subset viene assegnato alla variabile `subset2`.
4. **Crea il terzo subset**: Assegna le righe rimanenti nel DataFrame `df` al terzo subset, `subset3`.

Quindi, il codice suddivide il DataFrame `df` in tre subset di dimensioni simili.

1.1.23 Calcolo delle Percentuali delle Categorie nella Colonna 'class' del DataFrame

[23]:
```
percentuali_subset1 = subset1['class'].value_counts(normalize=True)
percentuali_subset1
```

[23]:
```
Compact Cars                    0.147906
Subcompact Cars                 0.125708
Midsize Cars                    0.117758
Standard Pickup Trucks          0.059824
Sport Utility Vehicle - 4WD     0.053999
Two Seaters                     0.048882
Large Cars                      0.048253
Sport Utility Vehicle - 2WD     0.043293
```

```
Small Station Wagons                  0.038334
Special Purpose Vehicles              0.037705
Minicompact Cars                      0.033297
Standard Pickup Trucks 2WD            0.031486
Vans                                  0.031250
Standard Pickup Trucks 4WD            0.025582
Midsize-Large Station Wagons          0.016924
Special Purpose Vehicle 2WD           0.016766
Midsize Station Wagons                0.014484
Small Pickup Trucks                   0.014405
Small Sport Utility Vehicle 4WD       0.014247
Small Pickup Trucks 2WD               0.012358
Standard Sport Utility Vehicle 4WD    0.011729
Vans, Cargo Type                      0.011178
Small Sport Utility Vehicle 2WD       0.010076
Minivan - 2WD                         0.008265
Special Purpose Vehicle 4WD           0.007714
Vans, Passenger Type                  0.006691
Small Pickup Trucks 4WD               0.005746
Standard Sport Utility Vehicle 2WD    0.004329
Minivan - 4WD                         0.001574
Special Purpose Vehicles/2wd          0.000079
Standard Pickup Trucks/2wd            0.000079
Vans Passenger                        0.000079
Name: class, dtype: float64
```

Il codice fa quanto segue:

1. **Calcola le percentuali delle categorie nella colonna 'class'**: Utilizza il metodo `value_counts(normalize=True)` sul DataFrame `subset1` per calcolare la frequenza relativa delle diverse categorie presenti nella colonna 'class'. Questo restituisce una serie in cui l'indice è il nome della categoria e il valore è la percentuale di quella categoria nel DataFrame. Questa serie viene assegnata alla variabile `percentuali_subset1`.

Quindi, il codice calcola le percentuali delle diverse categorie presenti nella colonna 'class' del DataFrame `subset1`.

1.1.24 Calcolo e Stampa delle Frequenze Relative delle Categorie nella Colonna 'class' del DataFrame

[24]:
```python
# Calcolare la frequenza delle diverse categorie
frequenze = df['class'].value_counts(normalize=True)

print(f"Frequenze relative delle categorie:\n{frequenze}")
```

```
Frequenze relative delle categorie:
Compact Cars                  0.144353
Subcompact Cars               0.130185
Midsize Cars                  0.110429
```

```
Standard Pickup Trucks                0.062495
Sport Utility Vehicle - 4WD           0.053207
Large Cars                            0.050610
Two Seaters                           0.049351
Sport Utility Vehicle - 2WD           0.042267
Small Station Wagons                  0.039827
Special Purpose Vehicles              0.039827
Minicompact Cars                      0.034475
Standard Pickup Trucks 2WD            0.031169
Vans                                  0.030854
Standard Pickup Trucks 4WD            0.025030
Midsize-Large Station Wagons          0.017159
Special Purpose Vehicle 2WD           0.014640
Midsize Station Wagons                0.014325
Small Pickup Trucks                   0.013774
Small Sport Utility Vehicle 4WD       0.013617
Vans, Cargo Type                      0.013381
Small Sport Utility Vehicle 2WD       0.011177
Standard Sport Utility Vehicle 4WD    0.011019
Small Pickup Trucks 2WD               0.010232
Minivan - 2WD                         0.009130
Vans, Passenger Type                  0.008186
Special Purpose Vehicle 4WD           0.008028
Small Pickup Trucks 4WD               0.005116
Standard Sport Utility Vehicle 2WD    0.004644
Minivan - 4WD                         0.001259
Special Purpose Vehicles/2wd          0.000079
Standard Pickup Trucks/2wd            0.000079
Special Purpose Vehicle               0.000079
Name: class, dtype: float64
```

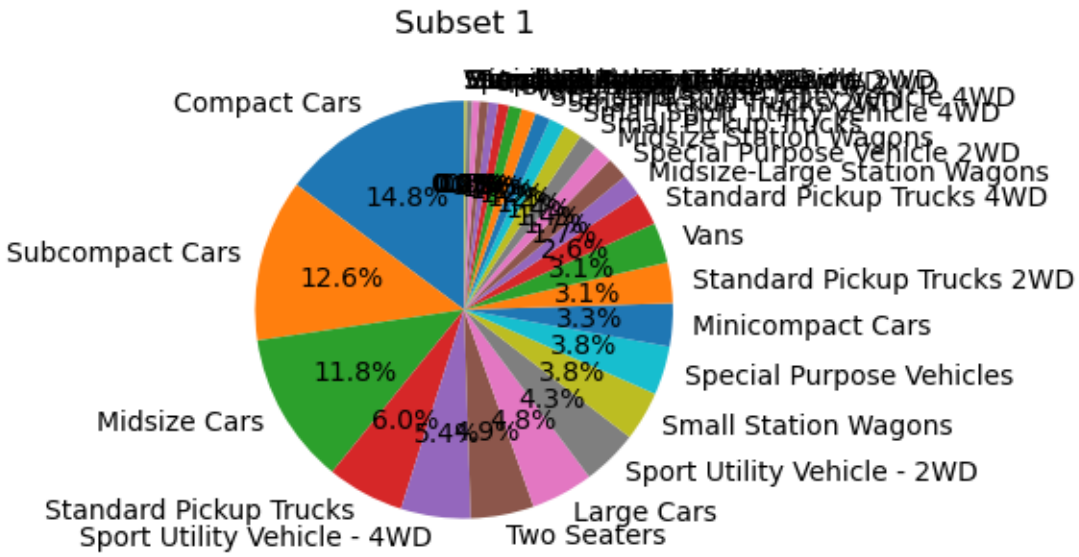
Il codice fa quanto segue:

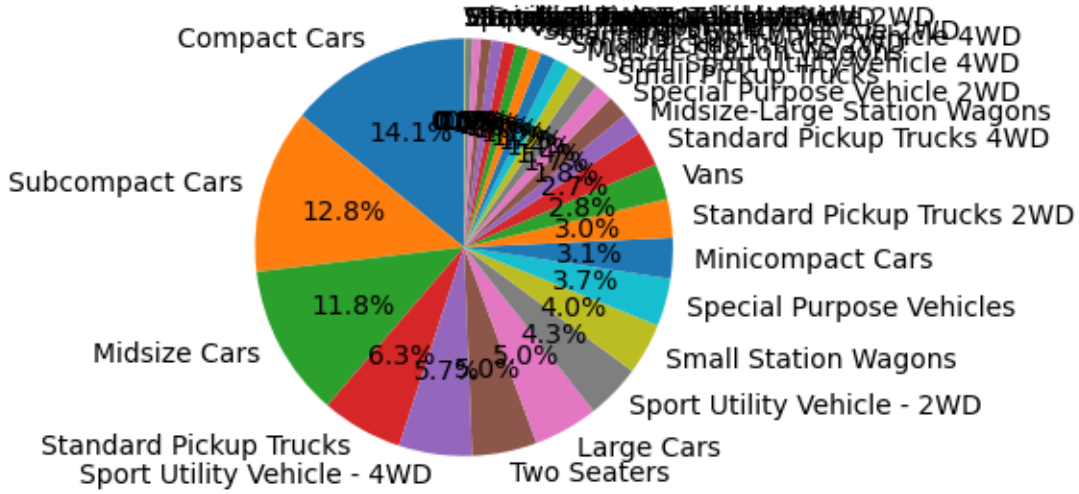
1. **Calcola le frequenze relative delle categorie**: Utilizza il metodo `value_counts(normalize=True)` sul DataFrame `df` per calcolare la frequenza relativa delle diverse categorie presenti nella colonna 'class'. Questo restituisce una serie in cui l'indice è il nome della categoria e il valore è la frequenza relativa di quella categoria nel DataFrame. Questa serie viene assegnata alla variabile `frequenze`.
2. **Stampa le frequenze relative delle categorie**: Utilizza il comando `print` per stampare le frequenze relative delle diverse categorie nella colonna 'class' del DataFrame `df`.

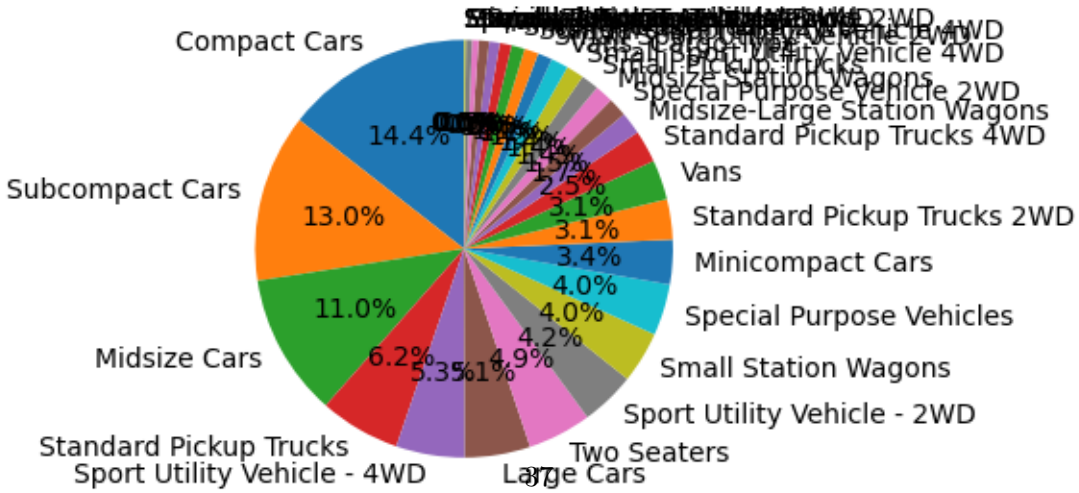
Quindi, il codice calcola e stampa le frequenze relative delle diverse categorie presenti nella colonna 'class' del DataFrame `df`.

1.1.25 Calcolo delle Percentuali delle Categorie per Ogni Subset e Creazione di Grafici a Torta

[25]:
```
percentuali_subset1 = subset1['class'].value_counts(normalize=True)
percentuali_subset2 = subset2['class'].value_counts(normalize=True)
percentuali_subset3 = subset3['class'].value_counts(normalize=True)

# Creare i grafici a torta
fig, axs = plt.subplots(3, 1, figsize=(6, 12))

# Subset 1
axs[0].pie(percentuali_subset1, labels=percentuali_subset1.index, autopct='%1.
 ↪1f%%', startangle=90)
axs[0].set_title('Subset 1')

# Subset 2
axs[1].pie(percentuali_subset2, labels=percentuali_subset2.index, autopct='%1.
 ↪1f%%', startangle=90)
axs[1].set_title('Subset 2')

# Subset 3
axs[2].pie(percentuali_subset3, labels=percentuali_subset3.index, autopct='%1.
 ↪1f%%', startangle=90)
axs[2].set_title('Subset 3')

# Mostrare il grafico
plt.show()
```


Subset 1
Compact Cars
Subcompact Cars
Midsize Cars
Standard Pickup Trucks
Sport Utility Vehicle - 4WD
Two Seaters
Large Cars
Sport Utility Vehicle - 2WD
Small Station Wagons
Special Purpose Vehicles
Minicompact Cars
Standard Pickup Trucks 2WD
Vans
Standard Pickup Trucks 4WD
Midsize-Large Station Wagons
Special Purpose Vehicle 2WD
Small Pickup Trucks
Midsize Station Wagons
14.8%
12.6%
11.8%
6.0%
5.4%
4.9%
4.8%
4.3%
3.8%
3.8%
3.3%
3.1%
3.1%
2.6%


Subset 2
Compact Cars
Subcompact Cars
Midsize Cars
Standard Pickup Trucks
Sport Utility Vehicle - 4WD
Two Seaters
Large Cars
Sport Utility Vehicle - 2WD
Small Station Wagons
Special Purpose Vehicles
Minicompact Cars
Standard Pickup Trucks 2WD
Vans
Standard Pickup Trucks 4WD
Midsize-Large Station Wagons
Special Purpose Vehicle 2WD
Small Pickup Trucks
Midsize Station Wagons
14.1%
12.8%
11.8%
6.3%
5.7%
5.0%
5.0%
4.3%
4.0%
3.7%
3.1%
3.0%
2.8%
2.7%


Subset 3
Compact Cars
Subcompact Cars
Midsize Cars
Standard Pickup Trucks
Sport Utility Vehicle - 4WD
Two Seaters
Large Cars
Sport Utility Vehicle - 2WD
Small Station Wagons
Special Purpose Vehicles
Minicompact Cars
Standard Pickup Trucks 2WD
Vans
Standard Pickup Trucks 4WD
Midsize-Large Station Wagons
Special Purpose Vehicle 2WD
Small Pickup Trucks
Midsize Station Wagons
14.4%
13.0%
11.0%
6.2%
5.3%
5.1%
4.9%
4.2%
4.0%
4.0%
3.4%
3.1%
3.1%
2.5%

Il codice fa quanto segue:

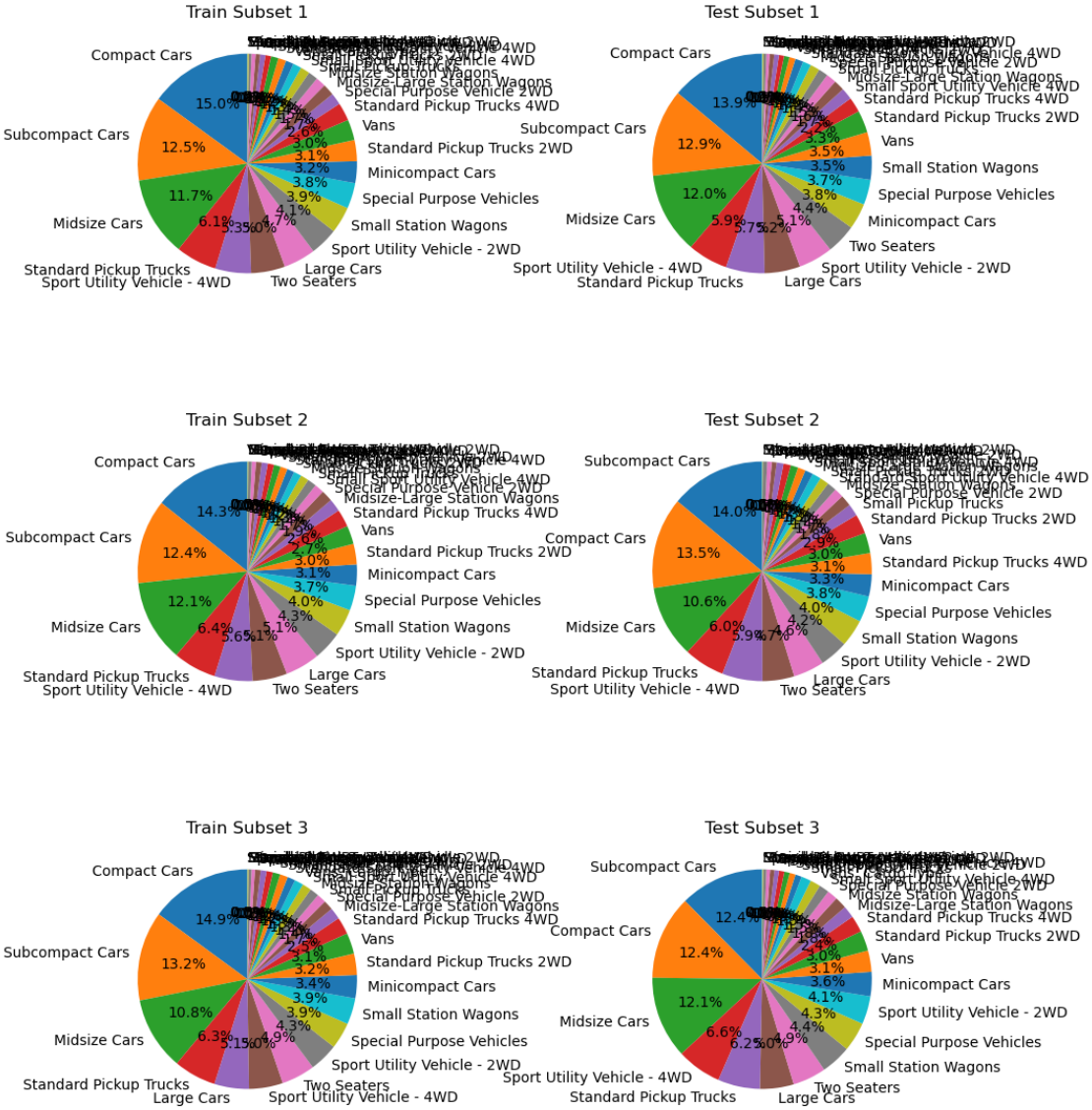
1. **Calcola le percentuali delle categorie nella colonna 'class' per ogni subset**: Utilizza il metodo `value_counts(normalize=True)` sui DataFrame `subset1`, `subset2` e `subset3` per calcolare la frequenza relativa delle diverse categorie presenti nella colonna 'class'. Questo restituisce tre serie in cui l'indice è il nome della categoria e il valore è la percentuale di quella categoria nel rispettivo subset. Queste serie vengono assegnate alle variabili `percentuali_subset1`, `percentuali_subset2` e `percentuali_subset3`.
2. **Imposta la dimensione del grafico**: Utilizza il metodo `subplots` di matplotlib.pyplot per creare una figura e un array di assi. La figura ha una dimensione di 6x12 e contiene tre assi disposti in una colonna.
3. **Crea i grafici a torta**: Utilizza il metodo `pie` su ciascuno degli assi per creare un grafico a torta delle percentuali delle categorie nella colonna 'class' per ciascuno dei subset. I grafici a torta mostrano le etichette delle categorie, le percentuali delle categorie formattate come stringhe con una cifra decimale e iniziano da un angolo di 90 gradi.
4. **Imposta i titoli dei grafici**: Imposta i titoli dei grafici a 'Subset 1', 'Subset 2' e 'Subset 3' con il metodo `set_title`.
5. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.

Quindi, il codice calcola le percentuali delle diverse categorie presenti nella colonna 'class' per ciascuno dei subset, crea un grafico a torta per ciascuno dei subset e visualizza i grafici.

1.1.26 Suddivisione dei Subset in Training Set e Test Set, Creazione di Grafici a Torta delle Frequenze Relative delle Categorie, e Visualizzazione della Figura

[26]:
```
# Dividere ciascun subset in training set e test set
train_subset1, test_subset1 = train_test_split(subset1, test_size=0.2,␣
 ↪random_state=42)
train_subset2, test_subset2 = train_test_split(subset2, test_size=0.2,␣
 ↪random_state=42)
train_subset3, test_subset3 = train_test_split(subset3, test_size=0.2,␣
 ↪random_state=42)

# Creare il grafico con 6 torte
fig, axs = plt.subplots(3, 2, figsize=(10, 12))

# Funzione per disegnare una torta con etichette
def draw_pie(ax, data, title):
    ax.pie(data, labels=data.index, autopct='%1.1f%%', startangle=90)
    ax.set_title(title)

# Prima riga di torte (Subset 1)
draw_pie(axs[0, 0], train_subset1['class'].value_counts(normalize=True), 'Train␣
 ↪Subset 1')
```

```
draw_pie(axs[0, 1], test_subset1['class'].value_counts(normalize=True), 'Test␣
 ↪Subset 1')

# Seconda riga di torte (Subset 2)
draw_pie(axs[1, 0], train_subset2['class'].value_counts(normalize=True), 'Train␣
 ↪Subset 2')
draw_pie(axs[1, 1], test_subset2['class'].value_counts(normalize=True), 'Test␣
 ↪Subset 2')

# Terza riga di torte (Subset 3)
draw_pie(axs[2, 0], train_subset3['class'].value_counts(normalize=True), 'Train␣
 ↪Subset 3')
draw_pie(axs[2, 1], test_subset3['class'].value_counts(normalize=True), 'Test␣
 ↪Subset 3')

# Regolare lo spaziamento tra i subplots
plt.tight_layout()

# Mostrare il grafico
plt.show()
```



Il codice fa quanto segue:

1. **Suddivide ciascun subset in training set e test set**: Utilizza la funzione `train_test_split` della libreria sklearn per suddividere ciascuno dei subset in un training set (80% dei dati) e un test set (20% dei dati). Questa suddivisione viene fatta in modo casuale, ma la casualità è controllata dal parametro `random_state=42` per garantire che i risultati siano riproducibili.

2. **Crea una figura con sei subplot**: Utilizza il metodo `subplots` di matplotlib.pyplot per creare una figura e un array di assi. La figura ha una dimensione di 10x12 e contiene sei assi disposti in tre righe e due colonne.

3. **Definisce una funzione per disegnare un grafico a torta**: Definisce una funzione `draw_pie` che prende come input un asse, dei dati e un titolo, e disegna un grafico a torta dei

dati sull'asse con il titolo specificato.

4. **Disegna sei grafici a torta**: Utilizza la funzione `draw_pie` per disegnare sei grafici a torta: uno per il training set e uno per il test set di ciascuno dei tre subset. I dati per i grafici a torta sono le frequenze relative delle diverse categorie nella colonna 'class' dei rispettivi set, calcolate con il metodo `value_counts(normalize=True)`.
5. **Regola lo spaziamento tra i subplot**: Utilizza il metodo `tight_layout` di matplotlib.pyplot per regolare automaticamente lo spaziamento tra i subplot in modo che si adattino bene nella figura.
6. **Mostra il grafico**: Infine, il grafico viene visualizzato con il metodo `show`.

Quindi, il codice suddivide ciascuno dei tre subset in un training set e un test set, crea una figura con sei grafici a torta che mostrano le frequenze relative delle diverse categorie nella colonna 'class' di ciascuno dei set, e visualizza la figura.

1.1.27 Identificazione degli Outliers nella Colonna 'engine_index' del DataFrame

[27]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Lista con outliers da entrambi i lati

# Calcola la media e la deviazione standard
mean_value = df['engine_index'].mean()
std_dev = df['engine_index'].std()

# Identifica gli outliers considerando ±3 sigma dalla media
outliers = df[(df['engine_index'] > mean_value + 3 * std_dev) |␣
 ↪(df['engine_index'] < mean_value - 3 * std_dev)]
outliers
```

[27]:

	vehicle_id	year	make	model	\
37	27725	1984	Audi	4000	
38	26609	1984	Audi	4000 S quattro	
40	26930	1984	Audi	5000S	
43	28001	1984	Audi	5000S	
47	28171	1984	Audi	5000S Wagon	
…	…	…	…	…	
16204	13678	1997	Subaru	Legacy Wagon	
16206	13681	1997	Subaru	Legacy Wagon AWD	
16207	13680	1997	Subaru	Legacy Wagon AWD	
16208	13682	1997	Subaru	Legacy Wagon AWD	
16209	13403	1997	Subaru	SVX AWD	

	class	drive	\
37	Subcompact Cars	NaN	
38	Subcompact Cars	NaN	

```
40                     Midsize Cars                          NaN
43                     Midsize Cars                          NaN
47           Midsize Station Wagons                          NaN
…                              …                          …
16204  Midsize-Large Station Wagons           Front-Wheel Drive
16206  Midsize-Large Station Wagons  4-Wheel or All-Wheel Drive
16207  Midsize-Large Station Wagons  4-Wheel or All-Wheel Drive
16208  Midsize-Large Station Wagons  4-Wheel or All-Wheel Drive
16209               Subcompact Cars  4-Wheel or All-Wheel Drive

            transmission transmission_type  engine_index engine_descriptor  \
37        Manual 5-Speed               NaN         64016    (FFS) CA model
38        Manual 5-Speed               NaN         64061             (FFS)
40     Automatic 3-Speed               NaN         64071             (FFS)
43     Automatic 3-Speed               NaN         64076    (FFS) CA model
47     Automatic 3-Speed               NaN         64076    (FFS) CA model
…                    …                …            …                …
16204  Automatic 4-Speed             CLKUP         66020             (FFS)
16206  Automatic 4-Speed             CLKUP         66030             (FFS)
16207     Manual 5-Speed               NaN         66020             (FFS)
16208     Manual 5-Speed               NaN         66030             (FFS)
16209  Automatic 4-Speed             CLKUP         66040             (FFS)

       …  hours_to_charge_ac_240v  composite_city_mpg composite_highway_mpg  \
37     …                      0.0                   0                    0
38     …                      0.0                   0                    0
40     …                      0.0                   0                    0
43     …                      0.0                   0                    0
47     …                      0.0                   0                    0
…    …                        …                   …                    …
16204  …                      0.0                   0                    0
16206  …                      0.0                   0                    0
16207  …                      0.0                   0                    0
16208  …                      0.0                   0                    0
16209  …                      0.0                   0                    0

       composite_combined_mpg range_ft1 city_range_ft1  highway_range_ft1  \
37                          0         0            0.0                0.0
38                          0         0            0.0                0.0
40                          0         0            0.0                0.0
43                          0         0            0.0                0.0
47                          0         0            0.0                0.0
…                         …       …            …                …
16204                       0         0            0.0                0.0
16206                       0         0            0.0                0.0
16207                       0         0            0.0                0.0
16208                       0         0            0.0                0.0
```

```
16209                      0         0            0.0                 0.0

       range_ft2  city_range_ft2  highway_range_ft2
37           NaN             0.0                0.0
38           NaN             0.0                0.0
40           NaN             0.0                0.0
43           NaN             0.0                0.0
47           NaN             0.0                0.0
…            …               …                  …
16204        NaN             0.0                0.0
16206        NaN             0.0                0.0
16207        NaN             0.0                0.0
16208        NaN             0.0                0.0
16209        NaN             0.0                0.0

[202 rows x 81 columns]
```

Il codice fa quanto segue:

1. **Calcola la media e la deviazione standard**: Utilizza i metodi `mean()` e `std()` sul DataFrame `df` per calcolare la media e la deviazione standard della colonna 'engine_index'. Questi valori vengono assegnati alle variabili `mean_value` e `std_dev`.
2. **Identifica gli outliers**: Utilizza un'operazione di indicizzazione booleana sul DataFrame `df` per selezionare le righe in cui il valore della colonna 'engine_index' è maggiore della media più tre volte la deviazione standard o minore della media meno tre volte la deviazione standard. Questo è un metodo comune per identificare gli outliers in un set di dati, noto come il metodo delle "tre sigma". Questi outliers vengono assegnati alla variabile `outliers`.

Quindi, il codice calcola la media e la deviazione standard della colonna 'engine_index' del DataFrame `df`, e identifica gli outliers in base a questi valori.

1.1.28 Calcolo della Media e della Deviazione Standard della Colonna 'engine_index' del DataFrame

[28]:
```
# Calcola la media e la deviazione standard
mean_value = df['engine_index'].mean()
std_dev = df['engine_index'].std()
std_dev
```

[28]:
```
17644.252756453876
```

Il codice fa quanto segue:

1. **Calcola la media**: Utilizza il metodo `mean()` sul DataFrame `df` per calcolare la media della colonna 'engine_index'. Questo valore viene assegnato alla variabile `mean_value`.
2. **Calcola la deviazione standard**: Utilizza il metodo `std()` sul DataFrame `df` per calcolare la deviazione standard della colonna 'engine_index'. Questo valore viene assegnato alla variabile `std_dev`.

Quindi, il codice calcola la media e la deviazione standard della colonna 'engine_index' del DataFrame `df`.

1.1.29 Calcolo del Numero di Outliers per Ogni Riga del DataFrame

[29]:
```
df['Num_Outliers'] = df.filter(like='Outlier_').sum(axis=1)
df
```

[29]:

	vehicle_id	year	make	model	class	drive	transmission	transmission_type	engine_index	engine_descriptor
0	26587	1984	Alfa Romeo	GT V6 2.5	Minicompact Cars	NaN	Manual 5-Speed	NaN	9001	(FFS)
4	27550	1984	AM General	DJ Po Vehicle 2WD	Special Purpose Vehicle 2WD	2-Wheel Drive	Automatic 3-Speed	NaN	1830	(FFS)
6	27549	1984	AM General	FJ8c Post Office	Special Purpose Vehicle 2WD	2-Wheel Drive	Automatic 3-Speed	NaN	1831	(FFS)
7	28425	1984	AM General	FJ8c Post Office	Special Purpose Vehicle 2WD	2-Wheel Drive	Automatic 3-Speed	NaN	1881	(FFS) CA model
9	28455	1984	American Motors Corporation	Eagle 4WD	Special Purpose Vehicle 4WD	4-Wheel or All-Wheel Drive	Automatic 3-Speed	NaN	1574	(FFS) CA model
…	…	…	…	…	…	…	…	…	…	…
38091	37553	2017	Volvo	S60 AWD	Compact Cars	All-Wheel Drive	Automatic (S8)	NaN	107	SIDI
38095	37552	2017	Volvo	S60 Inscription AWD	Compact Cars	All-Wheel Drive	Automatic (S8)	NaN	106	SIDI
38102	37562	2017	Volvo	V60 CC AWD	Small Station Wagons	All-Wheel Drive	Automatic (S8)	NaN	105	SIDI
38103	37560	2017	Volvo	V60 FWD	Small Station Wagons	Front-Wheel Drive	Automatic (S8)	NaN	88	SIDI
38105	38412	2017	Volvo	V90 CC AWD	Midsize Station Wagons	All-Wheel Drive	Automatic (S8)	NaN	115	SIDI

```
       …  composite_city_mpg  composite_highway_mpg composite_combined_mpg  \
0      …                   0                      0                      0
4      …                   0                      0                      0
6      …                   0                      0                      0
7      …                   0                      0                      0
9      …                   0                      0                      0
…    …                   …                      …                    …
38091  …                   0                      0                      0
38095  …                   0                      0                      0
38102  …                   0                      0                      0
38103  …                   0                      0                      0
38105  …                   0                      0                      0

       range_ft1 city_range_ft1 highway_range_ft1  range_ft2  city_range_ft2  \
0              0            0.0                0.0        NaN             0.0
4              0            0.0                0.0        NaN             0.0
6              0            0.0                0.0        NaN             0.0
7              0            0.0                0.0        NaN             0.0
9              0            0.0                0.0        NaN             0.0
…          …            …                …        …            …
38091          0            0.0                0.0        NaN             0.0
38095          0            0.0                0.0        NaN             0.0
38102          0            0.0                0.0        NaN             0.0
38103          0            0.0                0.0        NaN             0.0
38105          0            0.0                0.0        NaN             0.0

       highway_range_ft2  Num_Outliers
0                    0.0           0.0
4                    0.0           0.0
6                    0.0           0.0
7                    0.0           0.0
9                    0.0           0.0
…                  …           …
38091                0.0           0.0
38095                0.0           0.0
38102                0.0           0.0
38103                0.0           0.0
38105                0.0           0.0

[12705 rows x 82 columns]
```

Il codice fa quanto segue:

1. **Filtra le colonne che contengono 'Outlier_' nel loro nome**: Utilizza il metodo `filter(like='Outlier_')` sul DataFrame `df` per selezionare tutte le colonne il cui nome contiene la stringa 'Outlier_'.
2. **Calcola la somma lungo l'asse delle righe**: Utilizza il metodo `sum(axis=1)` sul risultato del filtro per calcolare la somma di ciascuna riga. Questo restituisce una serie in cui l'indice

è l'indice del DataFrame `df` e il valore è la somma dei valori nelle colonne filtrate per quella riga.

3. **Assegna la serie a una nuova colonna nel DataFrame**: Assegna la serie alla nuova colonna 'Num_Outliers' nel DataFrame `df`.

Quindi, il codice calcola il numero di outliers in ciascuna riga del DataFrame `df` (dove un outlier è definito come un valore in una colonna il cui nome contiene la stringa 'Outlier_') e assegna questi numeri a una nuova colonna 'Num_Outliers' nel DataFrame.

1.1.30 Calcolo del Numero di Caratteristiche nel DataFrame

[30]:
```
# Organizza i grafici in una matrice, con una colonna e 4 righe
num_features = len(df.columns) - 1  # Escludi la colonna 'Is_Outlier'
num_features
```

[30]:
```
81
```

Il codice fa quanto segue:

1. **Calcola il numero di caratteristiche (colonne)**: Utilizza la funzione `len(df.columns)` per ottenere il numero totale di colonne nel DataFrame `df`. Sottrae 1 da questo numero per escludere la colonna 'Is_Outlier'. Questo valore viene assegnato alla variabile `num_features`.

Quindi, il codice calcola il numero di caratteristiche (escludendo la colonna 'Is_Outlier') nel DataFrame `df`.