DR. ALVIN'S PUBLICATIONS

DATA CLEANSING A EUROPEAN AUTOMOBILE DATASET

WITH PYTHON

DR. ALVIN ANG

CONTENTS

I. STEP 1: PREVIEWING THE DATA

The file is here https://www.alvinang.sg/s/auto.csv

https://www.alvinang.sg/s/Data_Cleansing_a_European_Automobile_Dataset_with_Python_by_Dr_Alvin_Ang.ipynb

Realize that there are no headers!

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9	111	5000	21	27	13495
2	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9	111	5000	21	27	16500
3	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	six	152	mpfi	2.68	3.47	9	154	5000	19	26	16500
4	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	four	109	mpfi	3.19	3.4	10	102	5500	24	30	13950
5	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	five	136	mpfi	3.19	3.4	8	115	5500	18	22	17450
6	2	?	audi	gas	std	two	sedan	fwd	front	99.8	177.3	66.3	53.1	2507	ohc	five	136	mpfi	3.19	3.4	8.5	110	5500	19	25	15250
7	1	158	audi	gas	std	four	sedan	fwd	front	105.8	192.7	71.4	55.7	2844	ohc	five	136	mpfi	3.19	3.4	8.5	110	5500	19	25	17710
8	1	?	audi	gas	std	four	wagon	fwd	front	105.8	192.7	71.4	55.7	2954	ohc	five	136	mpfi	3.19	3.4	8.5	110	5500	19	25	18920
9	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	192.7	71.4	55.9	3086	ohc	five	131	mpfi	3.13	3.4	8.3	140	5500	17	20	23875
10	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	178.2	67.9	52	3053	ohc	five	131	mpfi	3.13	3.4	7	160	5500	16	22	?
11	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3	2395	ohc	four	108	mpfi	3.5	2.8	8.8	101	5800	23	29	16430
12	0	192	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3	2395	ohc	four	108	mpfi	3.5	2.8	8.8	101	5800	23	29	16925
13	0	188	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3	2710	ohc	six	164	mpfi	3.31	3.19	9	121	4250	21	28	20970
14	0	188	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3	2765	ohc	six	164	mpfi	3.31	3.19	9	121	4250	21	28	21105

A. IMPORT ALL LIBRARIES

Step 1: Previewing Data

1a): Import All Libraries

```
[25] import pandas as pd
     import missingno as msno
     import matplotlib.pylab as plt
     %matplotlib inline
```

B. INPUT HEADERS AND PREVIEW

1b) Load the Data and Preview

```
[26] headers = ["symboling",
              "normalized-losses",
              "make",
              "fuel-type",
              "aspiration",
              "num-of-doors",
              "body-style",
              "drive-wheels",
              "engine-location",
              "wheel-base",
              "length",
              "width",
              "height",
              "curb-weight",
              "engine-type",
              "num-of-cylinders",
              "engine-size",
              "fuel-system",
              "bore",
              "stroke",
              "compression-ratio",
              "horsepower",
              "peak-rpm",
              "city-mpg",
              "highway-mpg",
              "price"]

df = pd.read_csv("https://www.alvinang.sg/s/auto.csv", names = headers)

df.sample(5)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

II. STEP 2: VISUALIZING NANS

A. REPLACE "?" TO NAN

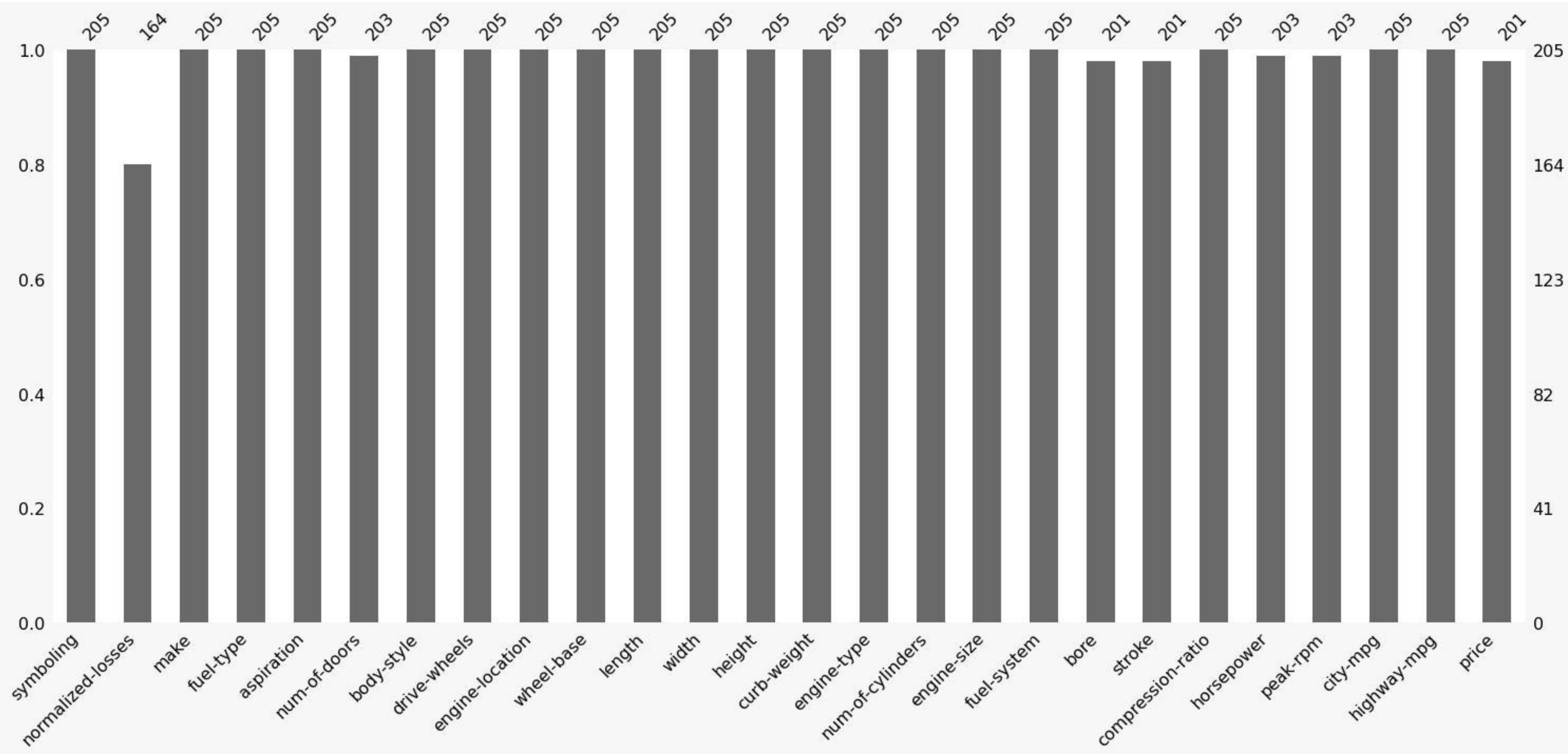
2a) Replace All '?' with NaNs

```
[27] import numpy as np

     # replace "?" to NaN
     df.replace("?", np.nan, inplace = True)

     df.head(5)
```

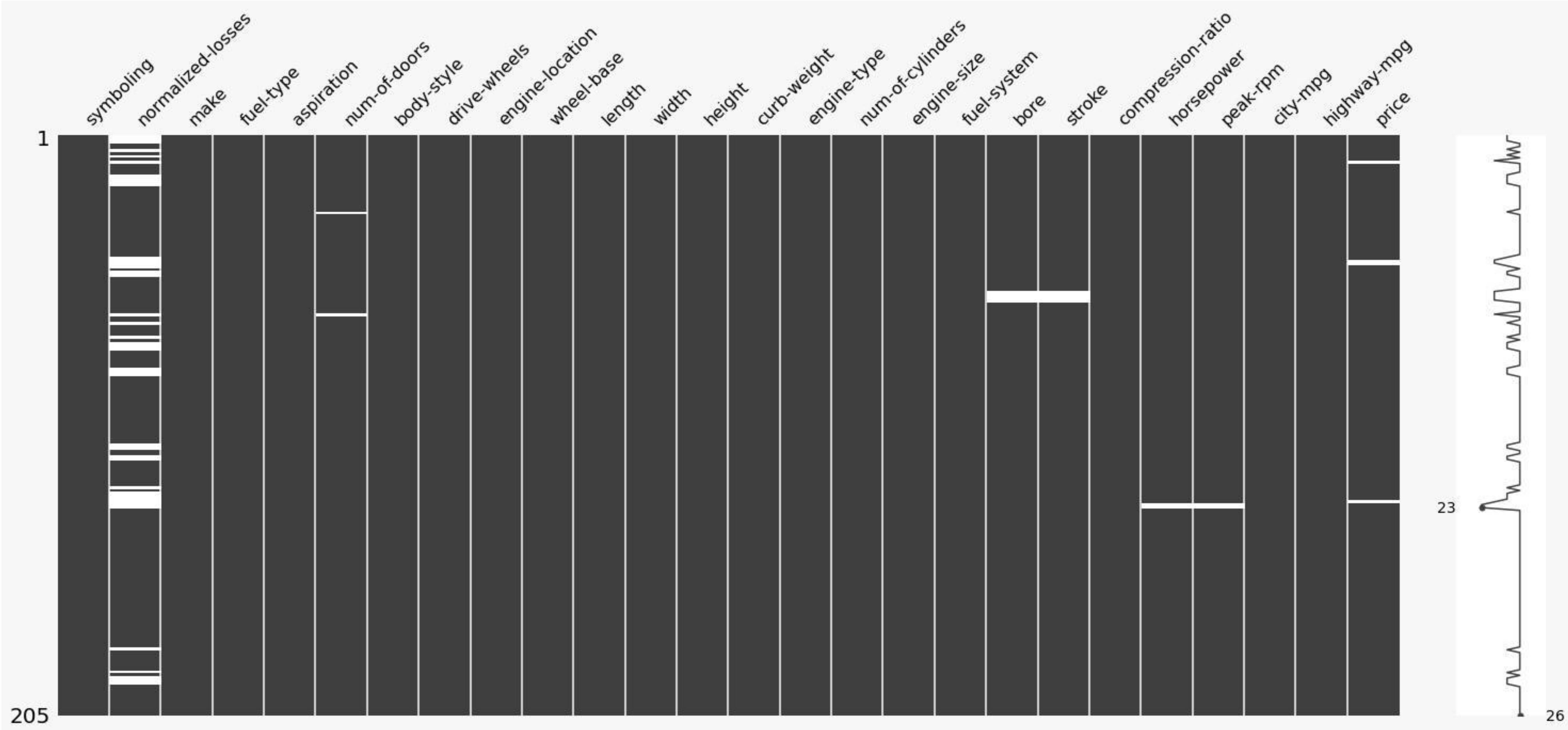
	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	
0	3	NaN	alfa-romero	gas	std	two	co
1	3	NaN	alfa-romero	gas	std	two	co
2	1	NaN	alfa-romero	gas	std	two	ha
3	2	164	audi	gas	std	four	
4	2	164	audi	gas	std	four	

2b) Using MissingNo to Visualize NaNs

```
msno.bar(df)
```



```
msno.matrix(df)
```



C. COUNT NUMBER OF NANS IN EACH COLUMN

2c) Count Number of NaNs in Each Column

```
[30] missing_data = df.isnull()
```

```
[31] for column in missing_data.columns.values.tolist():
       print(column)
       print(missing_data[column].value_counts())
       print("")
```

```
symboling
False    205
Name: symboling, dtype: int64

normalized-losses
False    164
True      41
Name: normalized-losses, dtype: int64

make
False    205
Name: make, dtype: int64

fuel-type
False    205
Name: fuel-type, dtype: int64

aspiration
False    205
Name: aspiration, dtype: int64
```

- Conclusion:

1. "normalized-losses": 41 missing data
2. "num-of-doors": 2 missing data
3. "bore": 4 missing data
4. "stroke" : 4 missing data
5. "horsepower": 2 missing data
6. "peak-rpm": 2 missing data
7. "price": 4 missing data

A. DROP ALL NANS IN THE PRICE COLUMN

Step 3: Dealing with Missing Data

3a) NaNs in the 'Price' Column - Drop All NaNs

```
[32] df.dropna(subset=["price"], axis=0, inplace=True)

     # "Price" column has 4 missing data
     # We simply delete the whole row
     # Reason: price is what we want to predict.
     # Any data entry without price data cannot be used for prediction;
     # therefore any row now without price data is not useful to us
     # Drop all rows with NaN values
```

- Finally, we have dropped all rows with missing values (in the 'price' column).

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9.0	111	5000	21	2	13495
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9.0	111	5000	21	2	16500
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	six	152	mpfi	2.68	3.47	9.0	154	5000	19	2	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	four	109	mpfi	3.19	3.40	10.0	102	5500	24	3	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	five	136	mpfi	3.19	3.40	8.0	115	5500	18	2	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
196	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	2952	ohc	four	141	mpfi	3.78	3.15	9.5	114	5400	23	2	16845
197	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.8	55.5	3049	ohc	four	141	mpfi	3.78	3.15	8.7	160	5300	19	2	19045
198	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3012	ohcv	six	173	mpfi	3.58	2.87	8.8	134	5500	18	2	21485
199	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3217	ohc	six	145	idi	3.01	3.40	23.0	106	4800	26	2	22470
200	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3062	ohc	four	141	mpfi	3.78	3.15	9.5	114	5400	19	2	22625

```
[33] df.reset_index(drop=True, inplace=True)

     # Reset the index because we dropped the rows
```

B. REPLACE ALL NANS IN THE NORMALIZED LOSSES COLUMN WITH THE MEAN

3b) NaNs in 'Normalized Losses' Column - Replace with Mean

```
[34] avg = df["normalized-losses"].astype("float").mean(axis=0)

     print("Average of Normalized-Losses Column:", avg)

     Average of Normalized-Losses Column: 122.0

[35] df["normalized-losses"].replace(np.nan, avg, inplace=True)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

	symboling	normalized-losses	make	fuel-type	aspi
0	3	122	alfa-romero	gas	
1	3	122	alfa-romero	gas	
2	1	122	alfa-romero	gas	
3	2	164	audi	gas	
4	2	164	audi	gas	
...	...	...	...	...	
196	-1	95	volvo	gas	
197	-1	95	volvo	gas	
198	-1	95	volvo	gas	
199	-1	95	volvo	diesel	

previously was NaN

now replaced with Mean

C. REPLACE ALL NANS IN THE NUMBER OF DOORS COLUMN WITH MOST OCCURING FREQUENCY

3c) NaNs in 'Number of Doors' Column - Replace with Most Occuring Frequency

```
[36] df['num-of-doors'].value_counts()

four    113
two      86
Name: num-of-doors, dtype: int64

[37] df['num-of-doors'].value_counts().idxmax()

'four'

[38] df["num-of-doors"].replace(np.nan, "four", inplace=True)
```

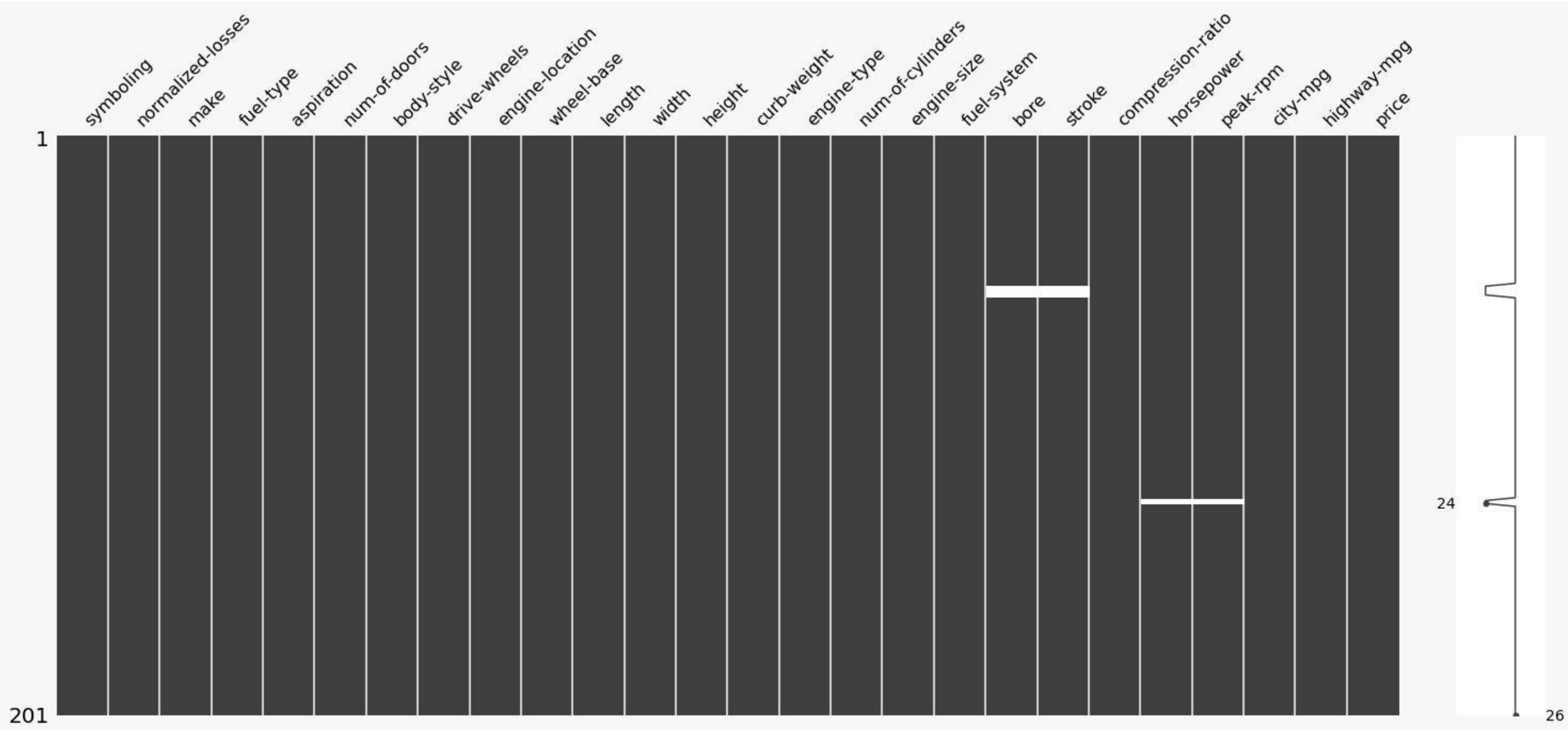
We see that the Most Frequently Occuring is Four Doors.

We replace all NaNs in that column to “Four” doors.

D. GLANCING AT ALL COLUMN NANS AGAIN

3d) Glancing at All Columns NaNs

```
[39] msno.matrix(df)

     #finally you see no more NaNs
```



Step 4: Changing the Columns Data Type

```
df.dtypes

#Normalized Losses is object = it should be integer
#Bore and Stroke are objects = they should be float
#Peak-rpm is object = it should be float
#Price is object = it should be float
```

```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

```
[ ] df[["normalized-losses"]] = df[["normalized-losses"]].astype("int")

    df[["bore", "stroke"]] = df[["bore", "stroke"]].astype("float")

    df[["peak-rpm"]] = df[["peak-rpm"]].astype("float")

    df[["price"]] = df[["price"]].astype("float")
```

Step 5: Transform Miles Per Gallon (MPG) to L/100km

```
df["highway-mpg"] = 235/df["highway-mpg"]
```

```
df.rename(columns={'highway-mpg':'highway-L/100km'}, inplace=True)
```

```
df['highway-L/100km'].head()

#we see that 'highway-mpg' has been renamed to 'highway-L/100km'
```

```
0     8.703704
1     8.703704
2     9.038462
3     7.833333
4    10.681818
Name: highway-L/100km, dtype: float64
```

Step 6: Normalizing Length + Width + Height

```
df['length'] = df['length']/df['length'].max()

df['width'] = df['width']/df['width'].max()

df['height'] = df['height']/df['height'].max()
```

```
df[["length","width","height"]].head()
```

	length	width	height
0	0.811148	0.890278	0.816054
1	0.811148	0.890278	0.816054
2	0.822681	0.909722	0.876254
3	0.848630	0.919444	0.908027
4	0.848630	0.922222	0.908027

A. MEANING OF NORMALIZING DATA

- Normalizing is to switch the scale to 0 to 1.
- Before Normalizing…

	length	width	height
0	168.8	64.1	0.816054
1	168.8	64.1	0.816054
2	171.2	65.5	0.876254
3	176.6	66.2	0.908027
4	176.6	66.4	0.908027

- After Normalizing….

	length	width	height
0	0.811148	0.890278	0.816054
1	0.811148	0.890278	0.816054
2	0.822681	0.909722	0.876254
3	0.848630	0.919444	0.908027
4	0.848630	0.922222	0.908027

VII. ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.