In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=pd.read_csv("cars.csv")
```

In [3]:
```
df
```

Out[3]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	13495
1	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	16500
2	1	?	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154	19	26	16500
3	2	164	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102	24	30	13950
4	2	164	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114	23	28	16845
201	-1	95	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160	19	25	19045
202	-1	95	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134	18	23	21485
203	-1	95	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106	26	27	22470
204	-1	95	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114	19	25	22625

205 rows × 15 columns

In [4]:
```
df.head()
```

Out[4]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	13495
1	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	16500
2	1	?	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154	19	26	16500
3	2	164	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102	24	30	13950
4	2	164	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115	18	22	17450

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    object 
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   body-style         205 non-null    object 
 5   drive-wheels       205 non-null    object 
 6   engine-location    205 non-null    object 
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object 
 10  engine-size        205 non-null    int64  
 11  horsepower         205 non-null    object 
 12  city-mpg           205 non-null    int64  
 13  highway-mpg        205 non-null    int64  
 14  price              205 non-null    int64  
dtypes: float64(2), int64(5), object(8)
memory usage: 24.1+ KB
```

In [6]:
```
df.describe()
```

Out[6]:

	symboling	width	height	engine-size	city-mpg	highway-mpg	price
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	65.907805	53.724878	126.907317	25.219512	30.751220	13227.478049
std	1.245307	2.145204	2.443522	41.642693	6.542142	6.886443	7902.651615
min	-2.000000	60.300000	47.800000	61.000000	13.000000	16.000000	5118.000000
25%	0.000000	64.100000	52.000000	97.000000	19.000000	25.000000	7788.000000
50%	1.000000	65.500000	54.100000	120.000000	24.000000	30.000000	10345.000000
75%	2.000000	66.900000	55.500000	141.000000	30.000000	34.000000	16500.000000
max	3.000000	72.300000	59.800000	326.000000	49.000000	54.000000	45400.000000

In [7]:
```
#change the data type of normalise-losses
df["normalized-losses"].replace("?",np.nan,inplace=True)
```

In [8]:
```
#change the data type of hoursepower
df["horsepower"].replace("?",np.nan,inplace=True)
```

In [9]:
```
df["normalized-losses"]=df["normalized-losses"].astype(float)
```

In [10]:
```
df.head()
```

Out[10]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	NaN	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	13495
1	3	NaN	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	16500
2	1	NaN	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115	18	22	17450

In [11]:
```
#replace the null value of normalized-losses by mean
nmean=df["normalized-losses"].mean()
df["normalized-losses"].fillna(nmean,inplace=True)
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
symboling           0
normalized-losses   0
make                0
fuel-type           0
body-style          0
drive-wheels        0
engine-location     0
width               0
height              0
engine-type         0
engine-size         0
horsepower          2
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

In [13]:
```
df.dropna()
```

Out[13]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114	23	28	16845
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160	19	25	19045
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134	18	23	21485
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106	26	27	22470
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114	19	25	22625

203 rows × 15 columns

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
symboling           0
normalized-losses   0
make                0
fuel-type           0
body-style          0
drive-wheels        0
engine-location     0
width               0
height              0
engine-type         0
engine-size         0
horsepower          2
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

In [15]:
```
df = df.dropna(how='all')
```

In [16]:
```
df
```

Out[16]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114	23	28	16845
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160	19	25	19045
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134	18	23	21485
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106	26	27	22470
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114	19	25	22625

205 rows × 15 columns

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

Out[17]:
```
symboling           0
normalized-losses   0
make                0
fuel-type           0
body-style          0
drive-wheels        0
engine-location     0
width               0
height              0
engine-type         0
engine-size         0
horsepower          2
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

In [18]:
```
f = df.dropna(thresh=len(df.columns)-1)
```

In [19]:
```
f
```

Out[19]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114	23	28	16845
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160	19	25	19045
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134	18	23	21485
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106	26	27	22470
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114	19	25	22625

205 rows × 15 columns

In [20]:
```
df.isnull().sum()
```

Out[20]:
```
symboling           0
normalized-losses   0
make                0
fuel-type           0
body-style          0
drive-wheels        0
engine-location     0
width               0
height              0
engine-type         0
engine-size         0
horsepower          2
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

In [21]:
```
df["horsepower"]=df["horsepower"].astype(float)
```

In [22]:
```
df
```

Out[22]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154.0	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102.0	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115.0	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	23	28	16845
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160.0	19	25	19045
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134.0	18	23	21485
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106.0	26	27	22470
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	19	25	22625

205 rows × 15 columns

In [23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    float64
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   body-style         205 non-null    object 
 5   drive-wheels       205 non-null    object 
 6   engine-location    205 non-null    object 
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object 
 10  engine-size        205 non-null    int64  
 11  horsepower         203 non-null    float64
 12  city-mpg           205 non-null    int64  
 13  highway-mpg        205 non-null    int64  
 14  price              205 non-null    int64  
dtypes: float64(4), int64(5), object(6)
memory usage: 24.1+ KB
```

In [24]:
```
df.isnull().sum()
```

Out[24]:
```
symboling            0
normalized-losses    0
make                 0
fuel-type            0
body-style           0
drive-wheels         0
engine-location      0
width                0
height               0
engine-type          0
engine-size          0
horsepower           2
city-mpg             0
highway-mpg          0
price                0
dtype: int64
```

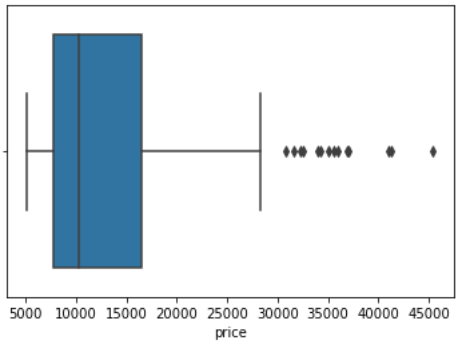
In [25]:
```
#replace the null value of normalized-losses by mean
nmean=df["horsepower"].mean()
df["horsepower"].fillna(nmean,inplace=True)
```

In [26]:
```
df.isnull().sum()
```

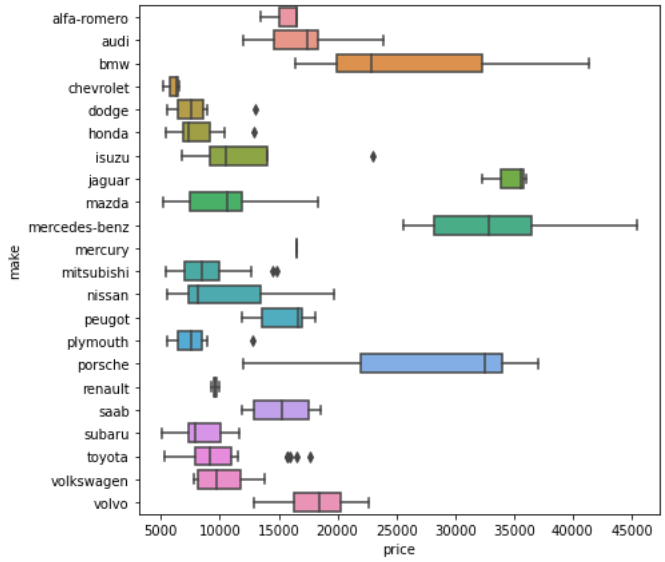
Out[26]:
```
symboling           0
normalized-losses   0
make                0
fuel-type           0
body-style          0
drive-wheels        0
engine-location     0
width               0
height              0
engine-type         0
engine-size         0
horsepower          0
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

In [27]:
```
#check the outlier and handle them
sns.boxplot(data=df,x="price")
```

Out[27]:
```
<AxesSubplot:xlabel='price'>
```



In [28]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x='price',y="make")
plt.show()
```



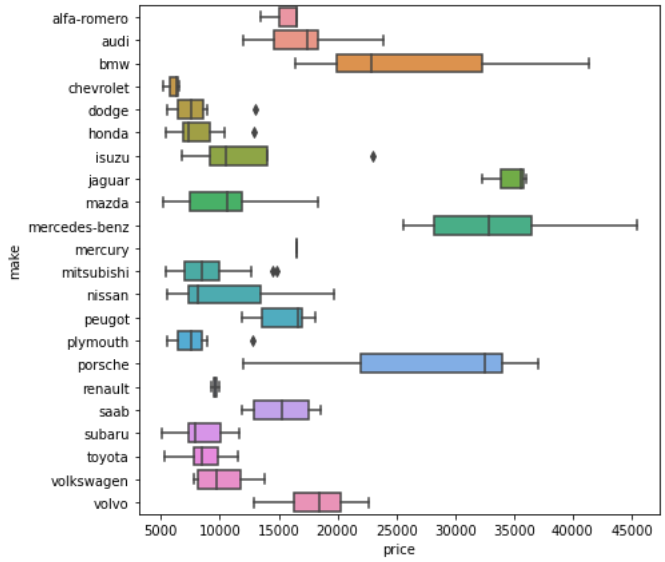
In [29]:
```
#removing ouliers of toyota
df[(df["make"]=="toyota")&(df["price"]>13000)]
```

Out[29]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
172	2	134.0	toyota	gas	convertible	rwd	front	65.6	53.0	ohc	146	116.0	24	30	17669
178	3	197.0	toyota	gas	hatchback	rwd	front	67.7	52.0	dohc	171	161.0	20	24	16558
179	3	197.0	toyota	gas	hatchback	rwd	front	67.7	52.0	dohc	171	161.0	19	24	15998
180	-1	90.0	toyota	gas	sedan	rwd	front	66.5	54.1	dohc	171	156.0	20	24	15690
181	-1	122.0	toyota	gas	wagon	rwd	front	66.5	54.1	dohc	161	156.0	19	24	15750

In [30]:
```
#removing ouliers of toyota
df.drop(index=[172,178,179,180,181],inplace=True)
```

In [31]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x='price',y="make")
plt.show()
```



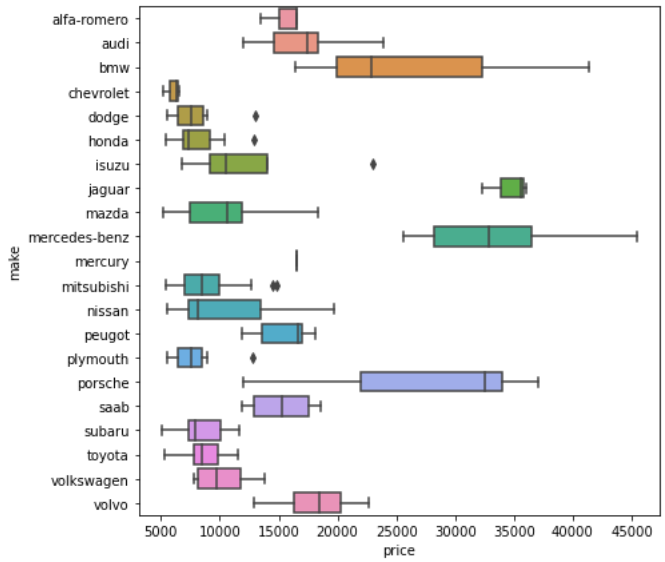
In [32]:
```
#removing ouliers   of
df[(df["make"]=="renault")&(df["price"]>1000)]
```

Out[32]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
130	0	122.0	renault	gas	wagon	fwd	front	66.5	55.2	ohc	132	104.256158	23	31	9295
131	2	122.0	renault	gas	hatchback	fwd	front	66.6	50.5	ohc	132	104.256158	23	31	9895

In [33]:
```
#removing ouliers of toyota
df.drop(index=[130,131],inplace=True)
```

In [34]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x='price',y="make")
plt.show()
```



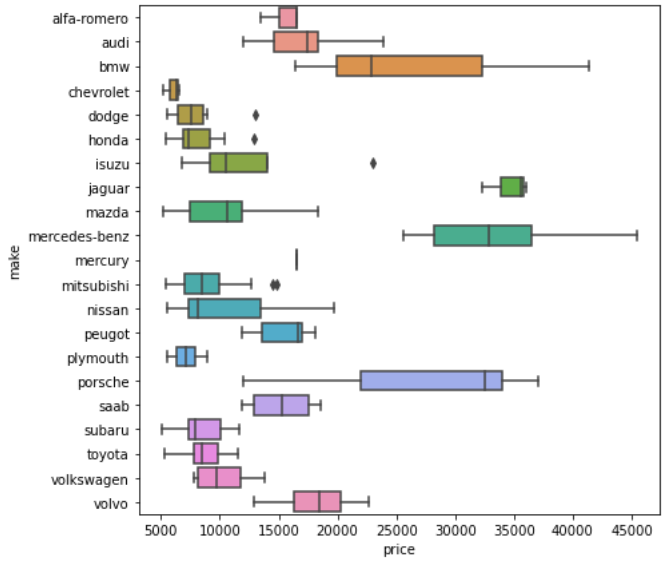
In [35]:
```
#removing ouliers plymouth   of
df[(df["make"]=="plymouth")&(df["price"]>12000)]
```

Out[35]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
124	3	122.0	plymouth	gas	hatchback	rwd	front	66.3	50.2	ohc	156	145.0	19	24	12764

In [36]:
```
#removing ouliers of toyota
df.drop(index=[124],inplace=True)
```

In [37]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x='price',y="make")
plt.show()
```



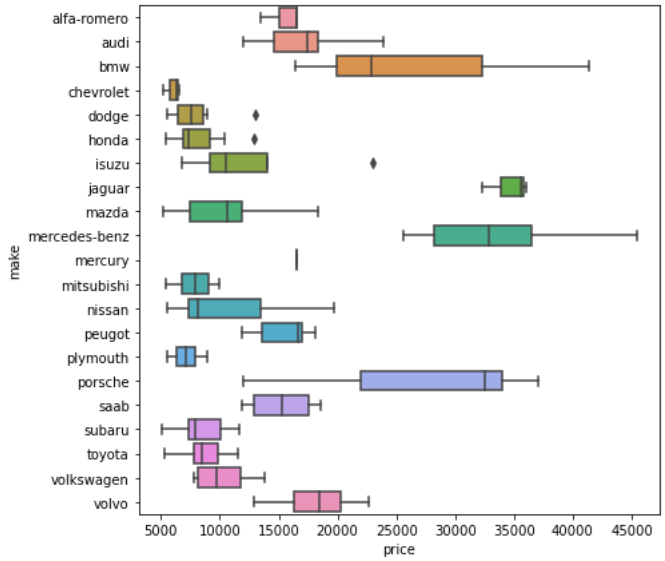
In [38]:
```
#removing ouliers mitsubshi   of
df[(df["make"]=="mitsubishi")&(df["price"]>12000)]
```

Out[38]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
82	3	122.0	mitsubishi	gas	hatchback	fwd	front	66.3	50.2	ohc	156	145.0	19	24	12629
83	3	122.0	mitsubishi	gas	hatchback	fwd	front	66.3	50.2	ohc	156	145.0	19	24	14869
84	3	122.0	mitsubishi	gas	hatchback	fwd	front	66.3	50.2	ohc	156	145.0	19	24	14489

In [39]:
```
#removing ouliers of mitsibishi
df.drop(index=[82,83,84],inplace=True)
```

In [40]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x='price',y="make")
plt.show()
```



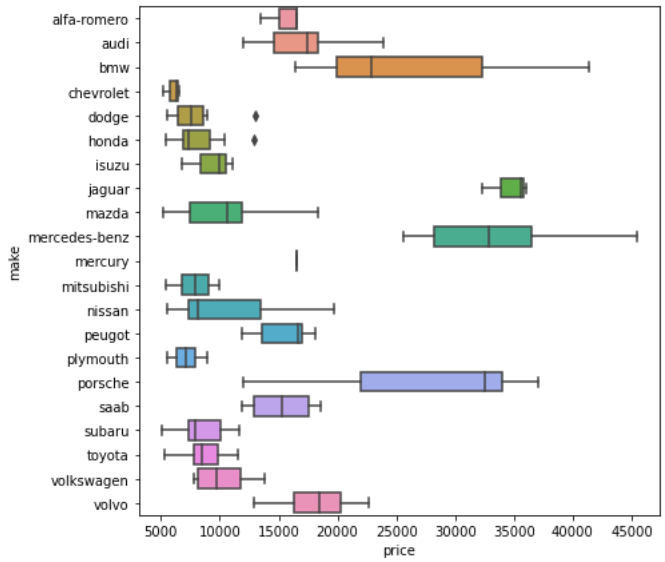
In [41]:
```
#removing ouliers isuzu  of
df[(df["make"]=="isuzu")&(df["price"]>13000)]
```

Out[41]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
45	0	122.0	isuzu	gas	sedan	fwd	front	63.6	52.0	ohc	90	70.0	38	43	23000

In [42]:
```
#removing ouliers of mitsubishi
df.drop(index=[45],inplace=True)
```

In [43]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x='price',y="make")
plt.show()
```



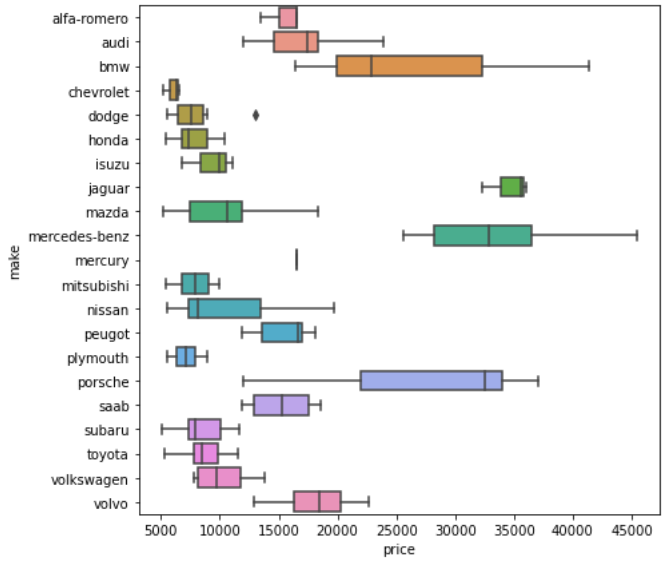
In [44]:
```
#removing ouliers honda  of
df[(df["make"]=="honda")&(df["price"]>12000)]
```

Out[44]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
41	0	85.0	honda	gas	sedan	fwd	front	65.2	54.1	ohc	110	101.0	24	28	12945

In [45]:
```
#removing ouliers of honda
df.drop(index=[41],inplace=True)
```

In [46]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x='price',y="make")
plt.show()
```



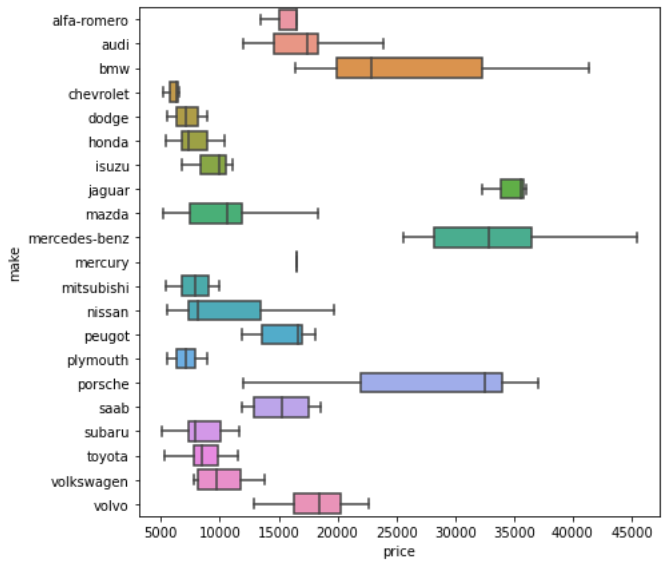
In [47]:
```
#removing ouliers dodge of
df[(df["make"]=="dodge")&(df["price"]>11000)]
```

Out[47]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
29	3	145.0	dodge	gas	hatchback	fwd	front	66.3	50.2	ohc	156	145.0	19	24	12964

In [48]:
```
#removing ouliers of dodge
df.drop(index=[29],inplace=True)
```

In [49]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x='price',y="make")
plt.show()
```



In [73]:
```
df_num = df.select_dtypes(["int64", "float64"])
df_cat = df.select_dtypes(object)
```

In [74]:
```
df_cat
```

Out[74]:

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
0	alfa-romero	gas	convertible	rwd	front	dohc
1	alfa-romero	gas	convertible	rwd	front	dohc
2	alfa-romero	gas	hatchback	rwd	front	ohcv
3	audi	gas	sedan	fwd	front	ohc
4	audi	gas	sedan	4wd	front	ohc
...	...	...	...	...	...	...
200	volvo	gas	sedan	rwd	front	ohc
201	volvo	gas	sedan	rwd	front	ohc
202	volvo	gas	sedan	rwd	front	ohcv
203	volvo	diesel	sedan	rwd	front	ohc
204	volvo	gas	sedan	rwd	front	ohc

191 rows × 6 columns

In [75]:
```
from sklearn.preprocessing import LabelEncoder
```

In [76]:
```
for col in df_cat:
 le = LabelEncoder()
 df_cat[col] = le.fit_transform(df_cat[col])
```

In [77]:
```
df_cat
```

Out[77]:

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
0	0	1	0	2	0	0
1	0	1	0	2	0	0
2	0	1	2	2	0	5
3	1	1	3	1	0	3
4	1	1	3	0	0	3
...	...	...	...	...	...	...
200	20	1	3	2	0	3
201	20	1	3	2	0	3
202	20	1	3	2	0	5
203	20	0	3	2	0	3
204	20	1	3	2	0	3

191 rows × 6 columns

In [79]:
```
df = pd.concat([df_cat, df_num], axis=1)
```

In [80]:
```
df.head()
```

Out[80]:

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg	price
0	0	1	0	2	0	0	3	122.0	64.1	48.8	130	111.0	21	27	13495
1	0	1	0	2	0	0	3	122.0	64.1	48.8	130	111.0	21	27	16500
2	0	1	2	2	0	5	1	122.0	65.5	52.4	152	154.0	19	26	16500
3	1	1	3	1	0	3	2	164.0	66.2	54.3	109	102.0	24	30	13950
4	1	1	3	0	0	3	2	164.0	66.4	54.3	136	115.0	18	22	17450

In [82]:
```
x = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [83]:
```
x
```

Out[83]:

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg
0	0	1	0	2	0	0	3	122.0	64.1	48.8	130	111.0	21	27
1	0	1	0	2	0	0	3	122.0	64.1	48.8	130	111.0	21	27
2	0	1	2	2	0	5	1	122.0	65.5	52.4	152	154.0	19	26
3	1	1	3	1	0	3	2	164.0	66.2	54.3	109	102.0	24	30
4	1	1	3	0	0	3	2	164.0	66.4	54.3	136	115.0	18	22
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	20	1	3	2	0	3	-1	95.0	68.9	55.5	141	114.0	23	28
201	20	1	3	2	0	3	-1	95.0	68.8	55.5	141	160.0	19	25
202	20	1	3	2	0	5	-1	95.0	68.9	55.5	173	134.0	18	23
203	20	0	3	2	0	3	-1	95.0	68.9	55.5	145	106.0	26	27
204	20	1	3	2	0	3	-1	95.0	68.9	55.5	141	114.0	19	25

191 rows × 14 columns

In [84]:
```
y
```

Out[84]:
```
0      13495
1      16500
2      16500
3      13950
4      17450
       ...
200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, Length: 191, dtype: int64
```

In [115]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.3, random_state=1)
```

In [116]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(xtrain, ytrain)
```

Out[116]:
```
LinearRegression()
```

In [117]:
```
train = linreg.score(xtrain, ytrain) # training acc
test = linreg.score(xtest, ytest) # testing acc
print(f"Traning Result -: {train}")
print(f"Test Result -: {test}")
```

```
Traning Result -: 0.9048915140262365
Test Result -: 0.7640523333560647
```

In [118]:
```
#High training acc and low testing acc.
#Low training error -: low bias
#high test error -: high variance
#overfitting
```

In [119]:
```
from sklearn.linear_model import Ridge, Lasso
```

In [120]:
```
#L2 regularization
```

In [121]:
```
l2=Ridge(alpha=5)
l2.fit(xtrain,ytrain)
```

Out[121]:
```
Ridge(alpha=5)
```

In [122]:
```
train = l2.score(xtrain, ytrain) # training acc
test = l2.score(xtest, ytest) # testing acc
print(f"Traning Result -: {train}")
print(f"Test Result -: {test}")
```

```
Traning Result -: 0.8881089825339492
Test Result -: 0.7248222410995846
```

In [127]:
```
#hypertuning lambda/ alpha value
for i in range(1,50):
 l2=Ridge(alpha=i)
 l2.fit(xtrain,ytrain)
 test = l2.score(xtest, ytest)
 print(f"value of lambda {i} test score {test}")
```

```
value of lambda 1 test score 0.7485738015381445
value of lambda 2 test score 0.738839703948545
value of lambda 3 test score 0.7325930354955605
value of lambda 4 test score 0.7281759053770177
value of lambda 5 test score 0.7248222410995846
value of lambda 6 test score 0.7221446635093223
value of lambda 7 test score 0.719927797725213
value of lambda 8 test score 0.718042246547955
value of lambda 9 test score 0.7164052810661192
value of lambda 10 test score 0.7149612769234575
value of lambda 11 test score 0.713671252802157
value of lambda 12 test score 0.7125069310070613
value of lambda 13 test score 0.7114471915250622
value of lambda 14 test score 0.7104758627424805
value of lambda 15 test score 0.7095802936457116
value of lambda 16 test score 0.7087504013397979
value of lambda 17 test score 0.7079780177819841
value of lambda 18 test score 0.7072564306425896
value of lambda 19 test score 0.7065800535191094
value of lambda 20 test score 0.7059441844149335
value of lambda 21 test score 0.7053448257419851
value of lambda 22 test score 0.7047785480391007
value of lambda 23 test score 0.7042423852976454
value of lambda 24 test score 0.7037337535045494
value of lambda 25 test score 0.70325038648876
value of lambda 26 test score 0.7027902848361902
value of lambda 27 test score 0.7023516747963539
value of lambda 28 test score 0.7019329749153014
value of lambda 29 test score 0.7015327687061176
value of lambda 30 test score 0.7011497820836137
value of lambda 31 test score 0.7007828645927239
value of lambda 32 test score 0.7004309736835657
value of lambda 33 test score 0.7000931614527095
value of lambda 34 test score 0.6997685633957174
value of lambda 35 test score 0.6994563888114121
value of lambda 36 test score 0.6991559125715555
value of lambda 37 test score 0.6988664680262362
value of lambda 38 test score 0.6985874408594415
value of lambda 39 test score 0.6983182637439611
value of lambda 40 test score 0.6980584116722383
value of lambda 41 test score 0.6978073978616419
value of lambda 42 test score 0.6975647701501456
value of lambda 43 test score 0.6973301078125644
value of lambda 44 test score 0.6971030187389297
value of lambda 45 test score 0.6968831369259654
value of lambda 46 test score 0.6966701202402779
value of lambda 47 test score 0.6964636484181885
value of lambda 48 test score 0.6962634212723827
value of lambda 49 test score 0.696069157079886
```

In [124]:
```
#final l2 model with best lambda value
```

In [125]:
```
l2=Ridge(alpha=11)
l2.fit(xtrain,ytrain)
```

Out[125]:
```
Ridge(alpha=11)
```

In [126]:
```
train = l2.score(xtrain, ytrain) # training acc
test = l2.score(xtest, ytest) # testing acc
print(f"Traning Result -: {train}")
print(f"Test Result -: {test}")
```

```
Traning Result -: 0.8828077338604756
Test Result -: 0.713671252802157
```

In [ ]:

In [ ]: