Lab3 – Descriptive Analysis

In this lab, you will learn some descriptive analysis methods to discover which features have the most impact on predict price. By the end of the lab you will learn to import data file into Jupyter Notebook, analyse individual feature using descriptive statistical analysis, visualization, correlation. The dataset (named as automobile.csv) is used for this lab.

Software: Jupyter Notebook

Procedure

1. In the desktop directory upload Lab 3 file (Lab3 – Descriptive Analysis – Participant Copy.ipynb) to the Jupyter Notebook.

2. In the desktop directory upload the data file (automobile.csv) to the Jupyter Notebook.

2. Click the Lab3 – Descriptive Analysis – Participant Copy.ipynb file to start the lab.

Import dataset

Import pandas and numpy libraries and load data to data frame as `df`.

In [ ]:

```
#Import pandas and numpy libraries

import pandas as pd

import numpy as np

#load data to dataframe as df

df = pd.read_csv('automobile.csv')

df.head()
```

Out [ ]:

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	...	9.0	154.0	5000.0	19	26
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	...	10.0	102.0	5500.0	24	30
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	...	8.0	115.0	5500.0	18	22

5 rows × 29 columns

In order to analyse individual feature using visualization, we need to install seaborn package.

In [ ]:
```
%%capture

! pip install seaborn
```

Exercise 3.1

Import matplotlib and seaborn When visualizing individual variables, it is important to understand the types of variables that we are dealing with. This will help us find the right visualization method for that variable.

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

import seaborn as sns

%matplotlib inline

#Write the code to list the data types for each column

#then press shift + enter
```

```
symboling              int64
normalized-losses      int64
make                  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                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                  int64
city-L/100km         float64
horsepower-binned     object
diesel                 int64
gas                    int64
dtype: object
```

Question 3.1

What is the data type of the feature `'compression-ratio'`?______________________

Descriptive Statistical Analysis

The `describe()` method is used to describe variables and NaN values are automatically skipped in this analysis. It includes the followings:

- count
- mean
- standard deviation (std)
- minimum value
- interquartile range: 25%, 50% and 75%
- maximum value

In [ ]:

```
df.describe()
```

Out[ ]:

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower
count	201.000000	201.00000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	197.000000	201.000000	201.000000
mean	0.840796	122.00000	98.797015	0.837102	0.915126	53.766667	2555.666667	126.875622	3.330692	3.256904	10.164279	103.405534
std	1.254802	31.99625	6.066366	0.059213	0.029187	2.447822	517.296727	41.546834	0.268072	0.319256	4.004965	37.365700
min	-2.000000	65.00000	86.600000	0.678039	0.837500	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000
25%	0.000000	101.00000	94.500000	0.801538	0.890278	52.000000	2169.000000	98.000000	3.150000	3.110000	8.600000	70.000000
50%	1.000000	122.00000	97.000000	0.832292	0.909722	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000
75%	2.000000	137.00000	102.400000	0.881788	0.925000	55.500000	2926.000000	141.000000	3.580000	3.410000	9.400000	116.000000
max	3.000000	256.00000	120.900000	1.000000	1.000000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	262.000000

The default setting of `describe` skips variables of type object. We can apply the method `describe` on the variables of type `object` as follows:

In [ ]:

```
df.describe(include=['object'])
```

Out[ ]:

	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	fuel-system	horsepower-binned
count	201	201	201	201	201	201	201	201	201	200
unique	22	2	2	5	3	2	6	7	8	3
top	toyota	std	four	sedan	fwd	front	ohc	four	mpfi	Low
freq	32	165	115	94	118	198	145	157	92	115

`value-counts` is used to find the number of units of each variable in the dataset. We can apply the `value_counts` method on the column `'drive-wheels'`. The method `value_counts` only works on Pandas series, not Pandas Data frames. As a result, we only include one bracket `df['drive-wheels']` not two brackets `df[['drive-wheels']]`. Then we convert it to data flame.

In [ ]:

```
drive_wheels_counts = df['drive-
wheels'].value_counts().to_frame()

drive_wheels_counts.rename(columns={'drive-wheels':
'value_counts'}, inplace=True)

drive_wheels_counts.index.name = 'drive-wheels'

drive_wheels_counts
```

Out[ ]:

	value_counts
drive-wheels	
fwd	118
rwd	75
4wd	8

Exercise 3.2

Write the code to display the value count with `'engine-location'` as variable.

In [ ]:

```
#Write the code and press shift + enter
```

Out[ ]:

engine-location	value_counts
front	198
rear	3

Question 3.2

What conclusion can we draw about the `'engine-location'`? Why?_______________

____

____

To visualize the mean price with `'drive-wheels'` and `'engine-location'` as pivot table.

In [ ]:

```
df_gptest = df[['drive-wheels','body-style','price']]

grouped_test1 = df_gptest.groupby(['drive-wheels','body-
style'],as_index=False).mean()

grouped_test1

grouped_pivot = grouped_test1.pivot(index='drive-
wheels',columns='body-style')

grouped_pivot = grouped_pivot.fillna(0)#fill missing values with 0

grouped_pivot
```

Out[ ]:

	price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	0.0	0.000000	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222

Correlation

We can calculate the correlation between variables of type `int64` or `float64` using the method `corr()`.

In [ ]:

```
df.corr()
```

Out[ ]:

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower	peak-
symboling	1.000000	0.466264	-0.535987	-0.365404	-0.242423	-0.550160	-0.233118	-0.110581	-0.140019	-0.008245	-0.182196	0.075819	0.279
normalized-losses	0.466264	1.000000	-0.056661	0.019424	0.086802	-0.373737	0.099404	0.112360	-0.029862	0.055563	-0.114713	0.217299	0.239
wheel-base	-0.535987	-0.056661	1.000000	0.876024	0.814507	0.590742	0.782097	0.572027	0.493244	0.158502	0.250313	0.371147	-0.360
length	-0.365404	0.019424	0.876024	1.000000	0.857170	0.492063	0.880665	0.685025	0.608971	0.124139	0.159733	0.579821	-0.285
width	-0.242423	0.086802	0.814507	0.857170	1.000000	0.306002	0.866201	0.729436	0.544885	0.188829	0.189867	0.615077	-0.245
height	-0.550160	-0.373737	0.590742	0.492063	0.306002	1.000000	0.307581	0.074694	0.180449	-0.062704	0.259737	-0.087027	-0.309
curb-weight	-0.233118	0.099404	0.782097	0.880665	0.866201	0.307581	1.000000	0.849072	0.644060	0.167562	0.156433	0.757976	-0.279
engine-size	-0.110581	0.112360	0.572027	0.685025	0.729436	0.074694	0.849072	1.000000	0.572609	0.209523	0.028889	0.822676	-0.256
bore	-0.140019	-0.029862	0.493244	0.608971	0.544885	0.180449	0.644060	0.572609	1.000000	-0.055390	0.001263	0.566936	-0.267
stroke	-0.008245	0.055563	0.158502	0.124139	0.188829	-0.062704	0.167562	0.209523	-0.055390	1.000000	0.187923	0.098462	-0.065
compression-ratio	-0.182196	-0.114713	0.250313	0.159733	0.189867	0.259737	0.156433	0.028889	0.001263	0.187923	1.000000	-0.214514	-0.435
horsepower	0.075819	0.217299	0.371147	0.579821	0.615077	-0.087027	0.757976	0.822676	0.566936	0.098462	-0.214514	1.000000	0.107
peak-rpm	0.279740	0.239543	-0.360305	-0.285970	-0.245800	-0.309974	-0.279361	-0.256733	-0.267392	-0.065713	-0.435780	0.107885	1.000
city-mpg	-0.035527	-0.225016	-0.470606	-0.665192	-0.633531	-0.049800	-0.749543	-0.650546	-0.582027	-0.034696	0.331425	-0.822214	-0.115
highway-mpg	0.036233	-0.181877	-0.543304	-0.698142	-0.680635	-0.104812	-0.794889	-0.679571	-0.591309	-0.035201	0.268465	-0.804575	-0.058
price	-0.082391	0.133999	0.584642	0.690628	0.751265	0.135486	0.834415	0.872335	0.543155	0.082310	0.071107	0.809575	-0.101
city-L/100km	0.066171	0.238567	0.476153	0.657373	0.673363	0.003811	0.785353	0.745059	0.554610	0.037300	-0.299372	0.889488	0.115
diesel	-0.196735	-0.101546	0.307237	0.211187	0.244356	0.281578	0.221046	0.070779	0.054458	0.241303	0.985231	-0.169053	-0.475

The diagonal elements are always one. We will study the Pearson correlation coefficient in the next lab.

Exercise 3.3

Write the code to find the correlation between the followings: `'bore'`, `'stroke'`, `'compression-ratio'` and `'horsepower'`.

In [ ]:

```
#Write the code and press shift + enter
```

Out[ ]:

	bore	stroke	compression-ratio	horsepower
bore	1.000000	-0.055390	0.001263	0.566936
stroke	-0.055390	1.000000	0.187923	0.098462
compression-ratio	0.001263	0.187923	1.000000	-0.214514
horsepower	0.566936	0.098462	-0.214514	1.000000

In order to understand the linear relationship between any feature and the `'price'`, we can use the `regplot()` which plots the scatterplot plus the fitted regression line for the data.

Positive linear relationship - Scatterplot of `'engine-size'` and `'price'`

In [ ]:

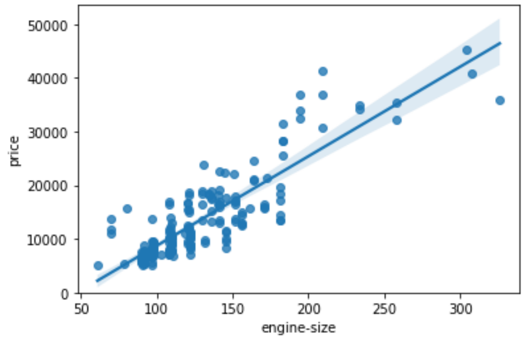
```
# Engine size as potential predictor variable of price

sns.regplot(x="engine-size", y="price", data=df)

plt.ylim(0,)
```

Out [ ]:

```
(0, 53570.72582628454)
```



This indicates that the `'engine-size'` is positively corelated to `'price'` as the `'engine-size'` goes up, the `'price'` goes up. After we perform the correlation between `'engine-size'` and `'price'`, the Pearson correlation coefficient is 0.87235.

In [ ]:

```
df[["engine-size", "price"]].corr()
```

Out [ ]:

	engine-size	price
engine-size	1.000000	0.872335
price	0.872335	1.000000

Negative linear relationship - Scatterplot of `'highway-mpg'` and `'price'`

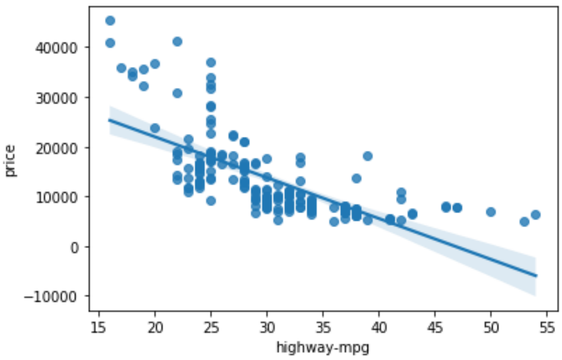
`'highway-mpg'` is also a potential predictor variable of `'price'`.

In [ ]:

```python
sns.regplot(x="highway-mpg", y="price", data=df)
```

Out [ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fa97d43808>
```



This indicates that the `'highway-mpg'` is negatively corelated to `'price'` as the `'highway-mpg'` goes up, the `'price'` goes down. After we perform the correlation between `'highway-mpg'` and `'price'`, the Pearson correlation coefficient is -0.704692.

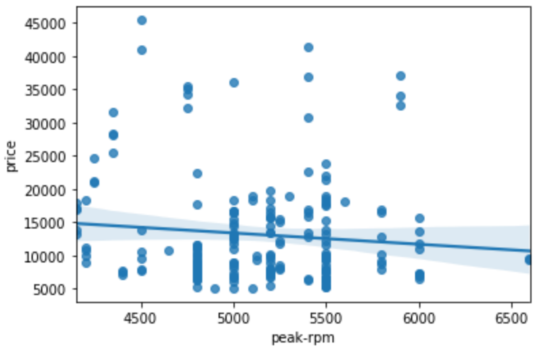
In [ ]:

```python
df[["highway-mpg", "price"]].corr()
```

Out [ ]:

	highway-mpg	price
highway-mpg	1.000000	-0.704692
price	-0.704692	1.000000

Weak or no correlation - Scatterplot of `'peak-rpm'` and `'price'`

In [ ]:	`sns.regplot(x="peak-rpm", y="price", data=df)`
Out [ ]:	<matplotlib.axes._subplots.AxesSubplot at 0x1fa97d3be88> 

`'peak-rpm'` is not a good predictor of the `'price'` as the regression line is close to horizontal. Moreover, the data are scattered and shows a lot of variability. The Pearson correlation coefficient for `'peak-rpm'` and `'price'` is -0.101616.

In [ ]:	`df[['peak-rpm','price']].corr()`
Out [ ]:	

	peak-rpm	price
peak-rpm	1.000000	-0.101616
price	-0.101616	1.000000

Exercise 3.4

Write the code to find the correlation between `'stroke'` vs `'price'`.

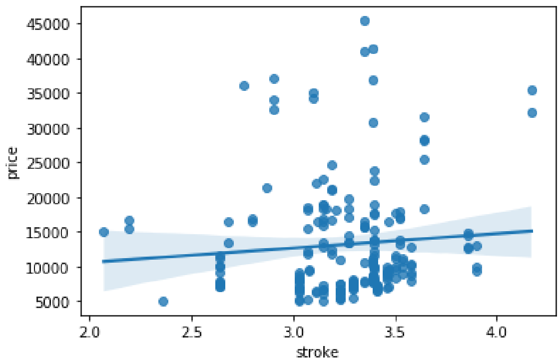
In [ ]:	`#Write the code and press shift + enter`
Out [ ]:	

	stroke	price
stroke	1.00000	0.08231
price	0.08231	1.00000

Question 3.3

What the correlation results between `'stroke'` and `'price'`?____________________
____
____
____

Write the code using *`regplot()`* to verify any linear relationship?

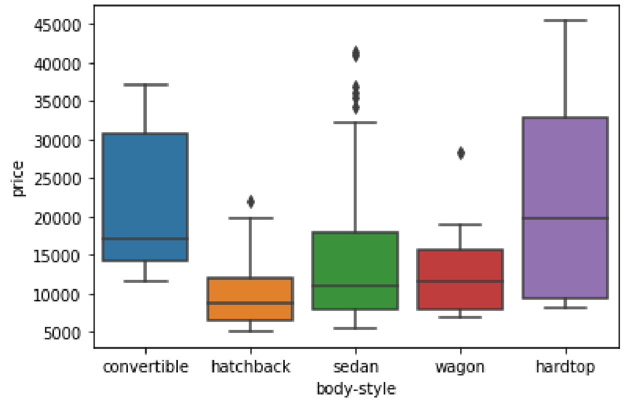
In [ ]:	*#Write the code and press shift + enter*
Out [ ]:	

Categorical Variables

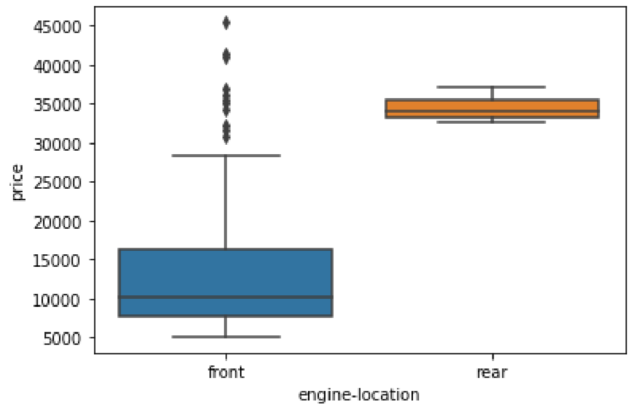
These are variables that describe the data unit with a small group of categories. The categorical variables can have the type `object` or `int64`. A good way to visualize categorical variables is by using *`boxplots()`*.

Let's use the *`boxplots()`* to see the relationship between `'body-style'` and `'price'`.

In [ ]:	*`sns.boxplot(x="body-style", y="price", data=df)`*

Out [ ]:	<matplotlib.axes._subplots.AxesSubplot at 0x203c0de51c8> 

Since the distribution of `'price'` and different `'body-style'` categories have a significantly overlap, so it is not a good predictor for `'price'`.

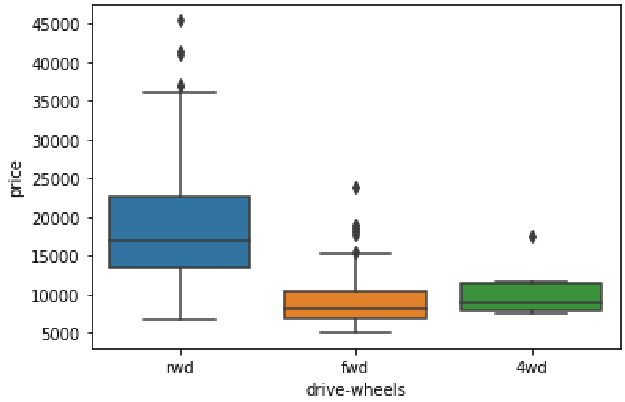
In [ ]:	`sns.boxplot(x="engine-location", y="price", data=df)`
Out [ ]:	<matplotlib.axes._subplots.AxesSubplot at 0x2b2834a38c8> 

Since the distribution of the price between the two engine locations (`front` and `rear`) are distinct enough, we can consider `'engine-location'` as a potential good predictor of `'price'`.

In [ ]:	`# boxplot() of drive-wheels and price`
	
	`sns.boxplot(x="drive-wheels", y="price", data=df)`

Out [ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x2b2834e1fc8>



Since the distribution of `'price'` between `'drive-wheels'` categories differ, `'drive-wheels'` could be a predictor of `'price'`.

After performing the descriptive analysis, we find that the following variables are import for predicting `'price'`:

- `'length'`
- `'width'`
- `'curb-weight'`
- `'engine-size'`
- `'horsepower'`
- `'city-mpg'`
- `'highway-mpg'`
- `'wheel-base'`
- `'bore'`
- `'drive-wheels'` (categorical variable)

--- End of Lab3 ---