Exploratory Data Analysis

September 19, 2022

[4]:
```
import numpy as np
import pandas as pd
# matplotlib = Basic Plots made easy
import matplotlib.pyplot as plt
# Descriptive Plots made easy
import seaborn as sns
```

[5]:
```
auto_price = pd.read_csv('https://raw.githubusercontent.com/ammishra08/
↪MachineLearning/master/Datasets/Automobile_price_data__Raw_.csv', sep = ',')
display(auto_price)
```

```
     symboling normalized-losses         make fuel-type aspiration  \
0            3                 ?  alfa-romero       gas        std
1            3                 ?  alfa-romero       gas        std
2            1                 ?  alfa-romero       gas        std
3            2               164         audi       gas        std
4            2               164         audi       gas        std
..         ...               ...          ...       ...        ...
200         -1                95        volvo       gas        std
201         -1                95        volvo       gas      turbo
202         -1                95        volvo       gas        std
203         -1                95        volvo    diesel      turbo
204         -1                95        volvo       gas      turbo

    num-of-doors   body-style drive-wheels engine-location  wheel-base  ...  \
0            two  convertible          rwd           front        88.6  ...
1            two  convertible          rwd           front        88.6  ...
2            two    hatchback          rwd           front        94.5  ...
3           four        sedan          fwd           front        99.8  ...
4           four        sedan          4wd           front        99.4  ...
..           ...          ...          ...             ...         ...  ...
200         four        sedan          rwd           front       109.1  ...
201         four        sedan          rwd           front       109.1  ...
202         four        sedan          rwd           front       109.1  ...
203         four        sedan          rwd           front       109.1  ...
204         four        sedan          rwd           front       109.1  ...

     engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
```

```
0            130         mpfi  3.47    2.68            9.0         111
1            130         mpfi  3.47    2.68            9.0         111
2            152         mpfi  2.68    3.47            9.0         154
3            109         mpfi  3.19    3.40           10.0         102
4            136         mpfi  3.19    3.40            8.0         115
..           ...          ...   ...     ...            ...         ...
200          141         mpfi  3.78    3.15            9.5         114
201          141         mpfi  3.78    3.15            8.7         160
202          173         mpfi  3.58    2.87            8.8         134
203          145          idi  3.01    3.40           23.0         106
204          141         mpfi  3.78    3.15            9.5         114

     peak-rpm city-mpg highway-mpg  price
0        5000       21          27  13495
1        5000       21          27  16500
2        5000       19          26  16500
3        5500       24          30  13950
4        5500       18          22  17450
..        ...      ...         ...    ...
200      5400       23          28  16845
201      5300       19          25  19045
202      5500       18          23  21485
203      4800       26          27  22470
204      5400       19          25  22625

[205 rows x 26 columns]
```

[6]:
```
auto_price.dtypes
```

[6]:
```
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
```

[7]:
```
# convert object columns to numerical columns
cols = ['bore','stroke','horsepower','peak-rpm','price']
auto_price[cols] = auto_price[cols].apply(pd.to_numeric, args = ('coerce',))
```

[8]:
```
auto_price.isnull().sum()
```

[8]:
```
symboling           0
normalized-losses   0
make                0
fuel-type           0
aspiration          0
num-of-doors        0
body-style          0
drive-wheels        0
engine-location     0
wheel-base          0
length              0
width               0
height              0
curb-weight         0
engine-type         0
num-of-cylinders    0
engine-size         0
fuel-system         0
bore                4
stroke              4
compression-ratio   0
horsepower          2
peak-rpm            2
city-mpg            0
highway-mpg         0
price               4
dtype: int64
```

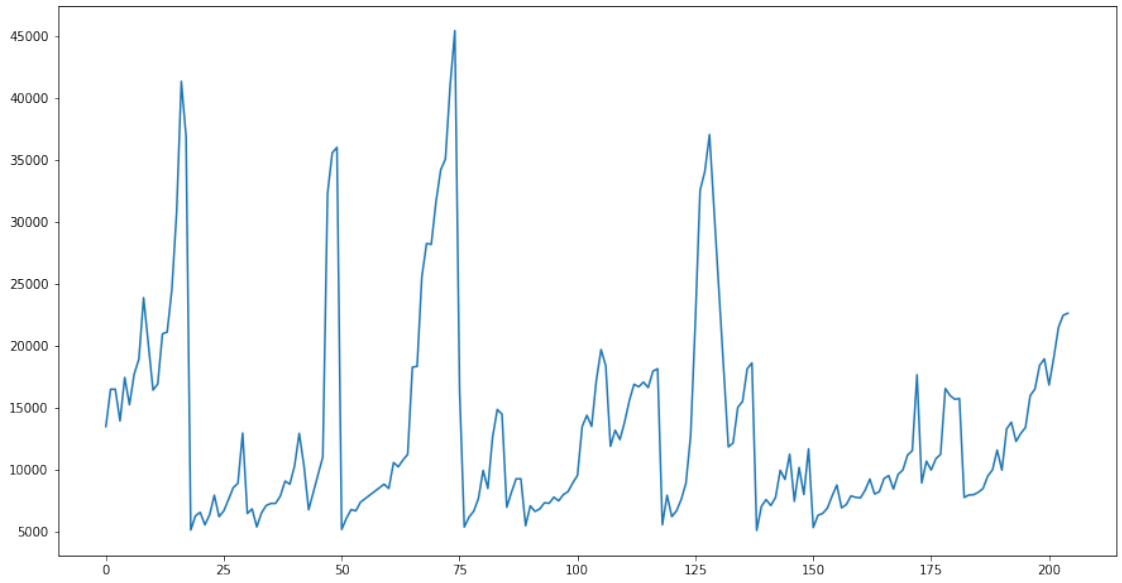
[9]:
```
# Remove/Drop the Missing Value Rows
auto_price.dropna(inplace = True)
```

Line Plot

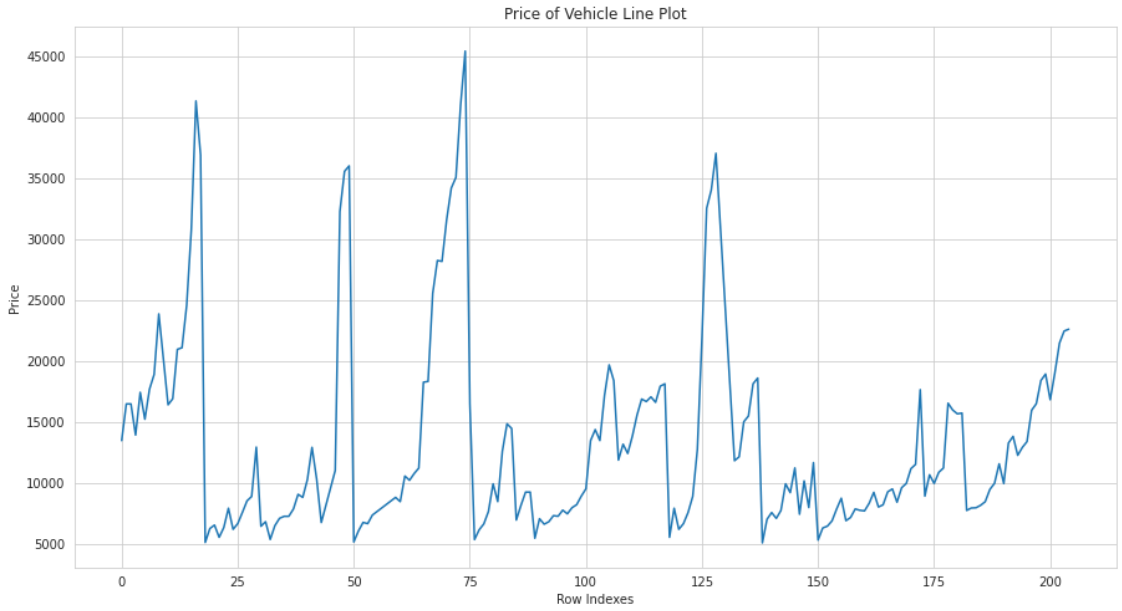
* Relationship Plot (Bivariate)
* Trends/Values - Univariate

[10]:
```
# Using Pandas for Visualization
auto_price['price'].plot(figsize = (15,8))
```

[10]: <AxesSubplot:>



[11]:
```
sns.set_style('whitegrid')
plt.figure(figsize = (15,8))
plt.plot(auto_price['price'])
plt.title('Price of Vehicle Line Plot', fontsize = 12)
plt.xlabel('Row Indexes')
plt.ylabel('Price')
plt.show()
```

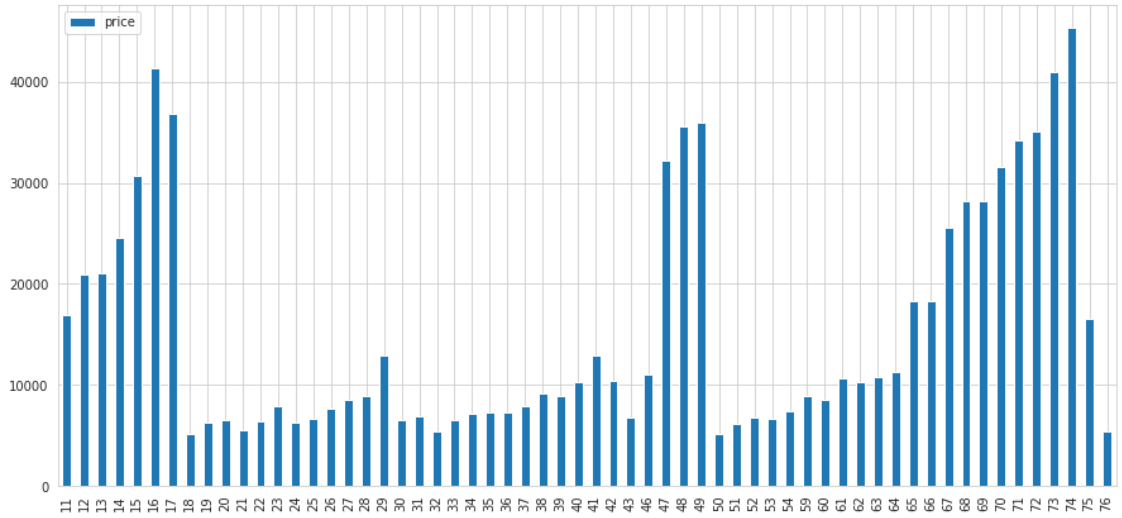


0.0.1 Bar Plot

- Bivariate Chart
- Bar chart used by categorical, nominal, Categorical Vs Numerical/Continous
- Bar plots are used to display counts of unique value of categorical data types, height of the bar represents count for each category

[12]:
```
sns.set_style('whitegrid')
auto_price[['price']][10:70].plot.bar(figsize = (15,7))
```

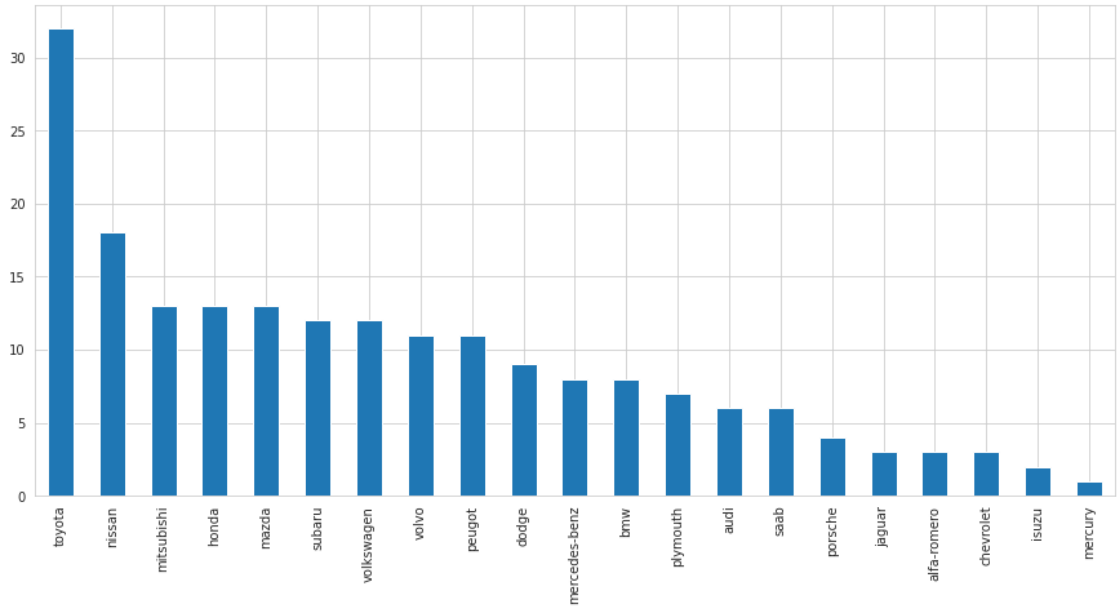
[12]: <AxesSubplot:>



[13]:
```
counts = auto_price['make'].value_counts()
counts
```

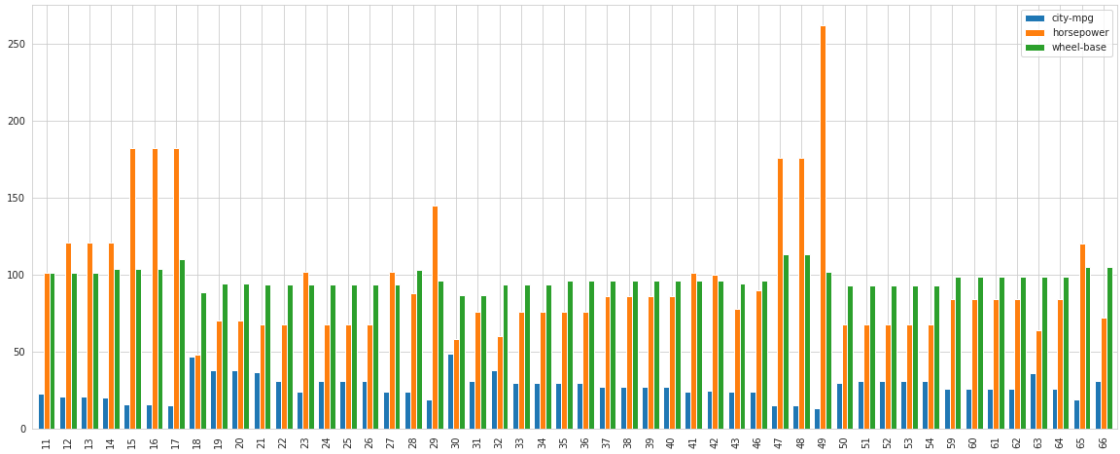
[13]:
```
toyota           32
nissan           18
mitsubishi       13
honda            13
mazda            13
subaru           12
volkswagen       12
volvo            11
peugot           11
dodge             9
mercedes-benz     8
bmw               8
plymouth          7
audi              6
saab              6
porsche           4
jaguar            3
alfa-romero       3
chevrolet         3
isuzu             2
mercury           1
Name: make, dtype: int64
```

[14]:
```
fig = plt.figure(figsize = (7,6))
auto_price['make'].value_counts().plot.bar(figsize = (15,7))
plt.show()
```



[15]:
```
auto_price[['city-mpg','horsepower','wheel-base']][10:60].plot.bar(figsize =␣
↪(20,8), width = 0.8)
```

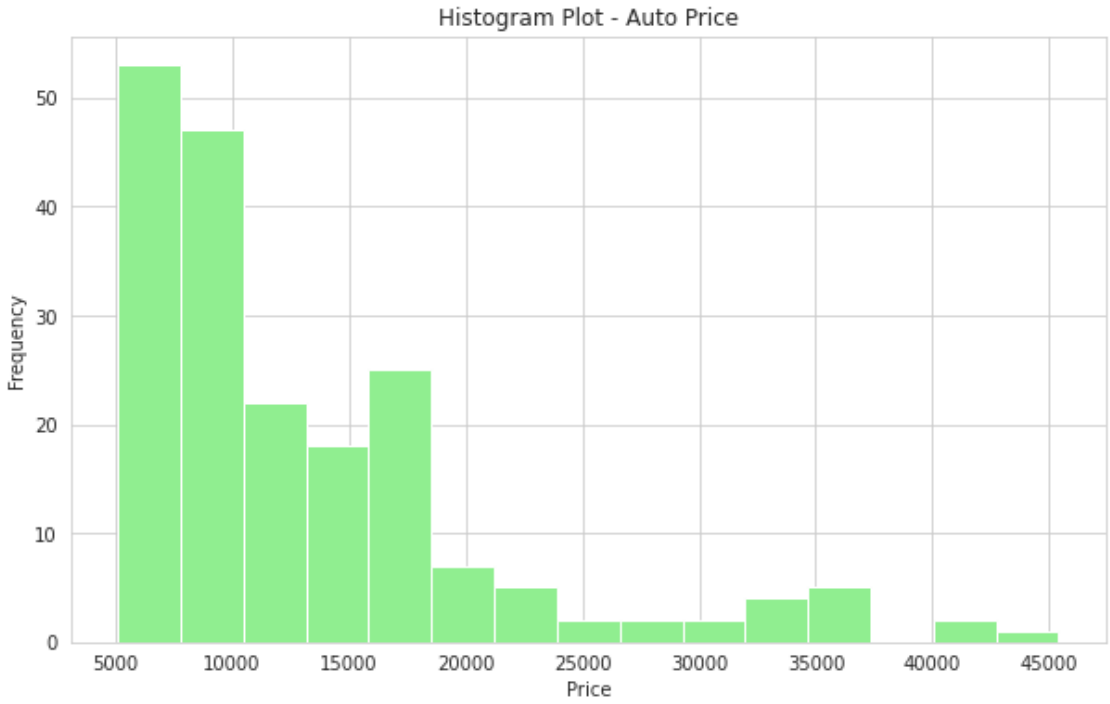
[15]: <AxesSubplot:>



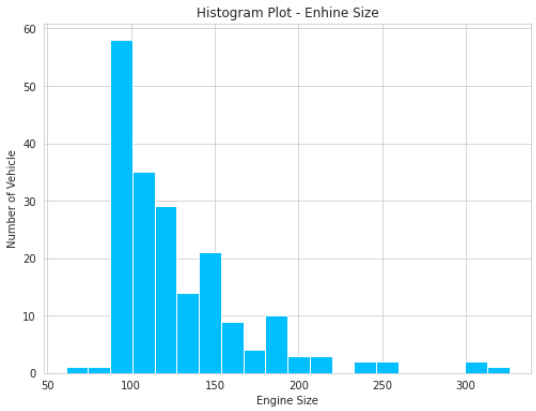
0.0.2 Histogram

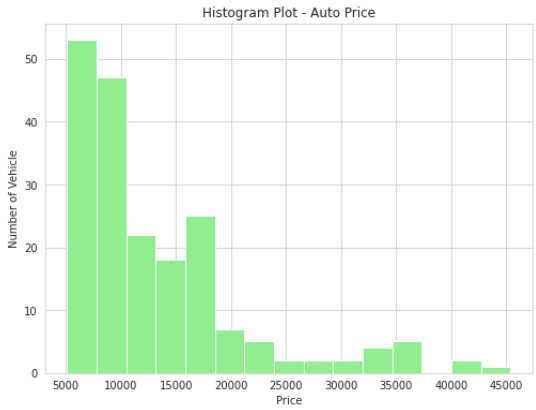
- Continuous samples - study the spread/distribution of data
- Univariate Analysis

[16]:
```
fig = plt.figure(figsize = (10,6))
# bins = intervals
plt.hist(auto_price['price'], color = 'lightgreen', bins = 15)
plt.title('Histogram Plot - Auto Price')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

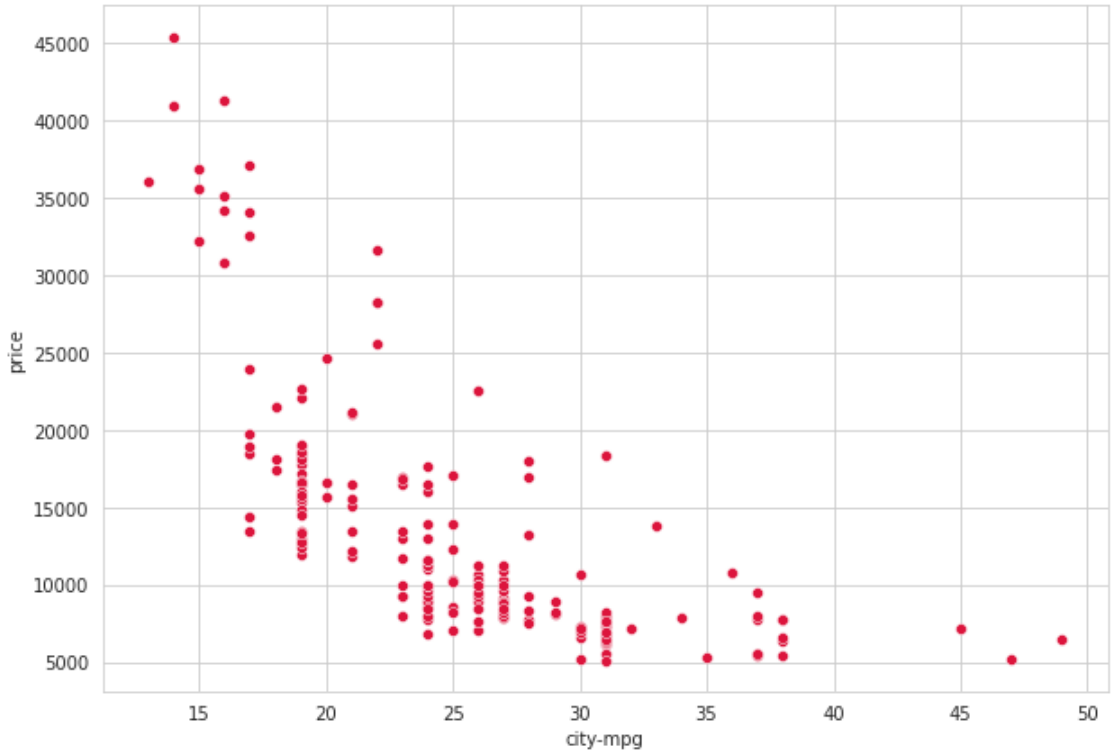


[17]:
```
# Subplots = (1, 2, 1) => 1st Row, 2nd Column, 1st Index
plt.figure(figsize = (18,6))
plt.subplot(1, 2, 1)
plt.hist(auto_price['engine-size'], color = 'deepskyblue', bins = 20)
plt.title('Histogram Plot - Enhine Size')
plt.xlabel('Engine Size')
plt.ylabel('Number of Vehicle')
plt.subplot(1, 2, 2)
plt.hist(auto_price['price'], color = 'lightgreen', bins = 15)
plt.title('Histogram Plot - Auto Price')
plt.xlabel('Price')
plt.ylabel('Number of Vehicle')
plt.show()
```



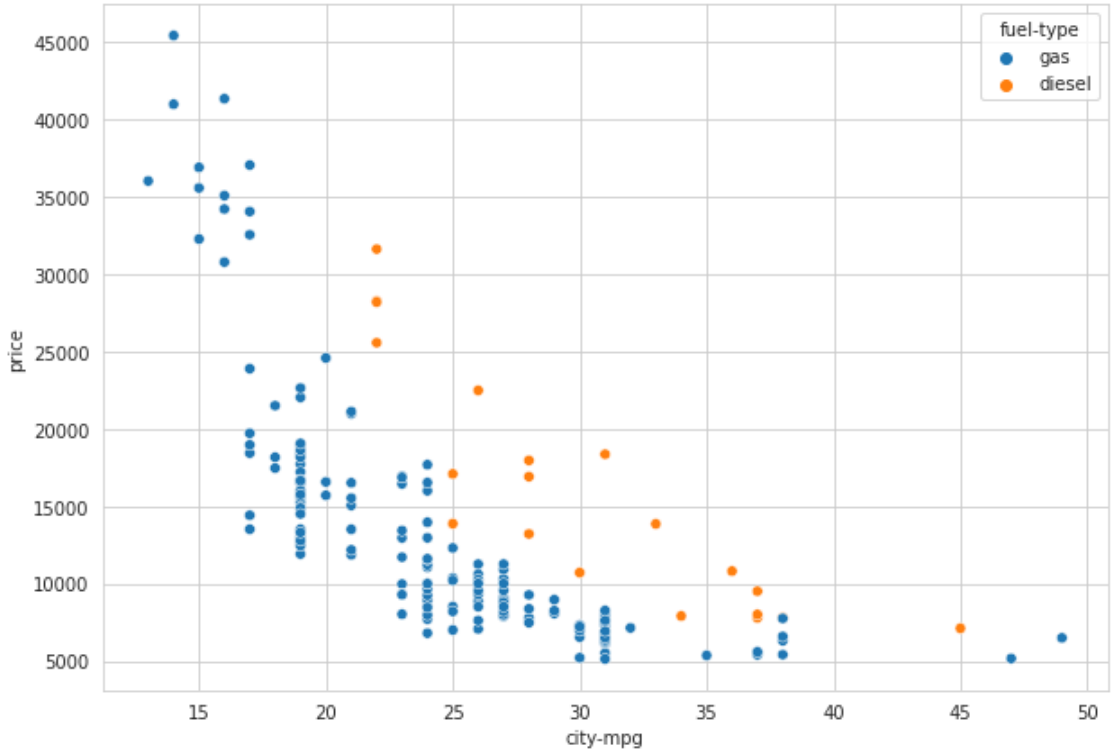


[18]:
```
plt.figure(figsize = (10,7))
sns.scatterplot(x = auto_price['city-mpg'], y = auto_price['price'], color =␣
↪'crimson')
plt.show()
```



[19]:
```
plt.figure(figsize = (10,7))
# marker = changing the scatter plots style o,x,X,s,d,>,<,*,.,^
```

```
sns.scatterplot(x = auto_price['city-mpg'], y = auto_price['price'], hue =␣
↪auto_price['fuel-type'], color = 'crimson')
plt.show()
```



Scatter Plot

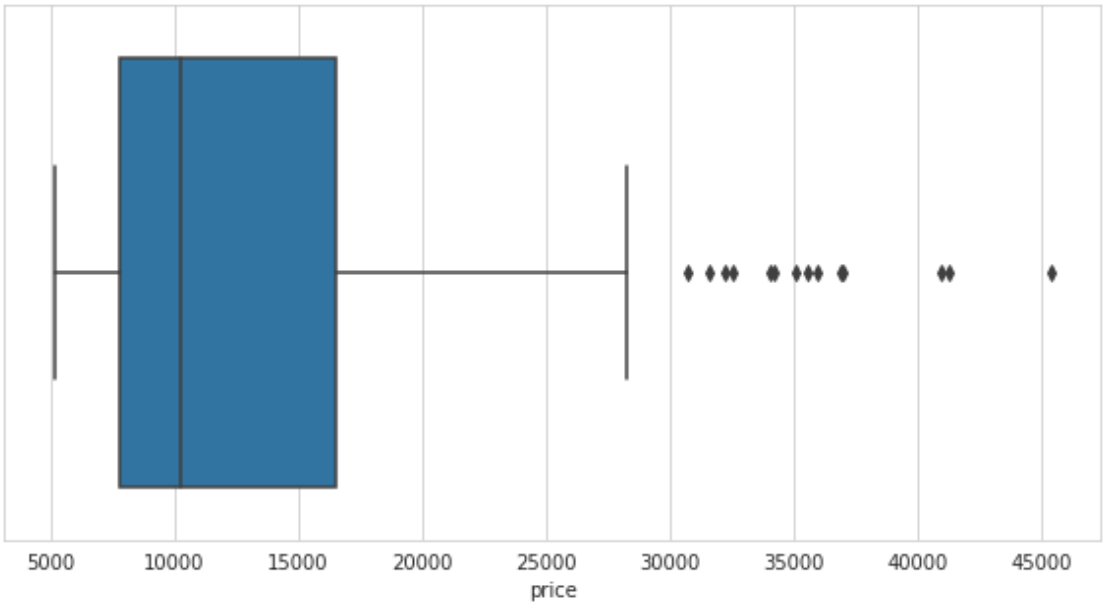
```
* Scatter Plot is plotted between numerical values.
* Shows relationship b/w x & y plots
```

0.0.3 Box Plot

- Distribution of sample data
- Five point summary - min, max, Q1, Q2, median
- Outliers

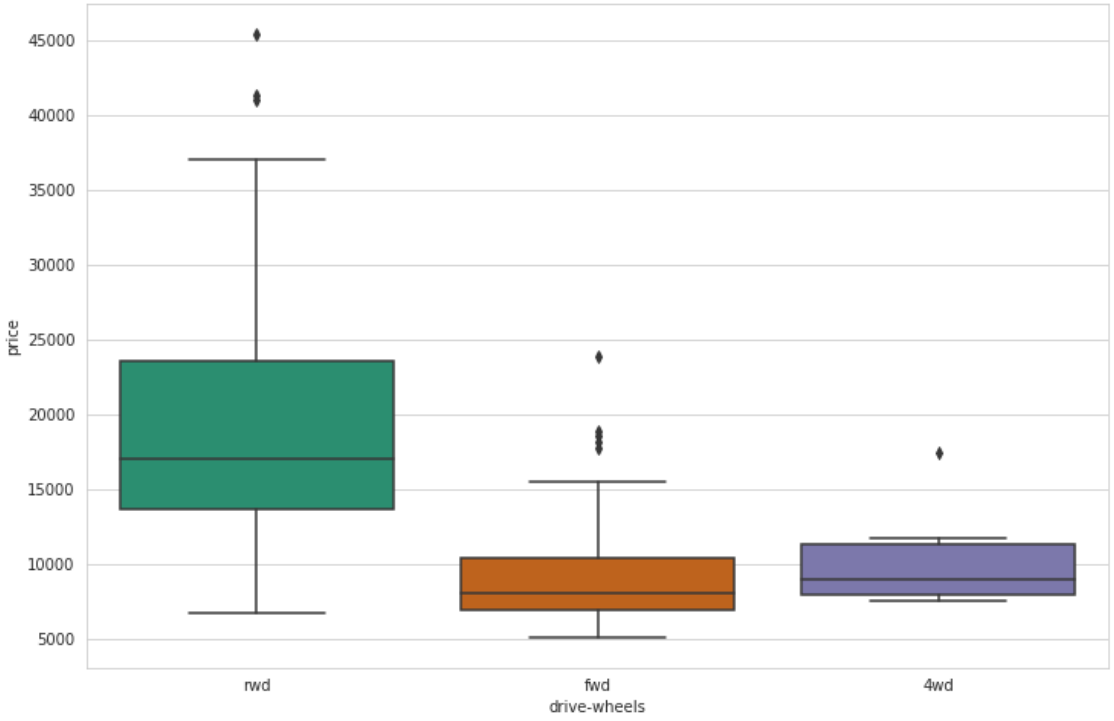
[20]:
```
plt.figure(figsize = (10,5))
sns.boxplot(x = 'price', data = auto_price)
```

[20]:
```
<AxesSubplot:xlabel='price'>
```



[21]:
```
plt.figure(figsize = (12,8))
sns.boxplot(x = 'drive-wheels', y = 'price', data = auto_price, palette =␣
↪'Dark2')
```

[21]: <AxesSubplot:xlabel='drive-wheels', ylabel='price'>

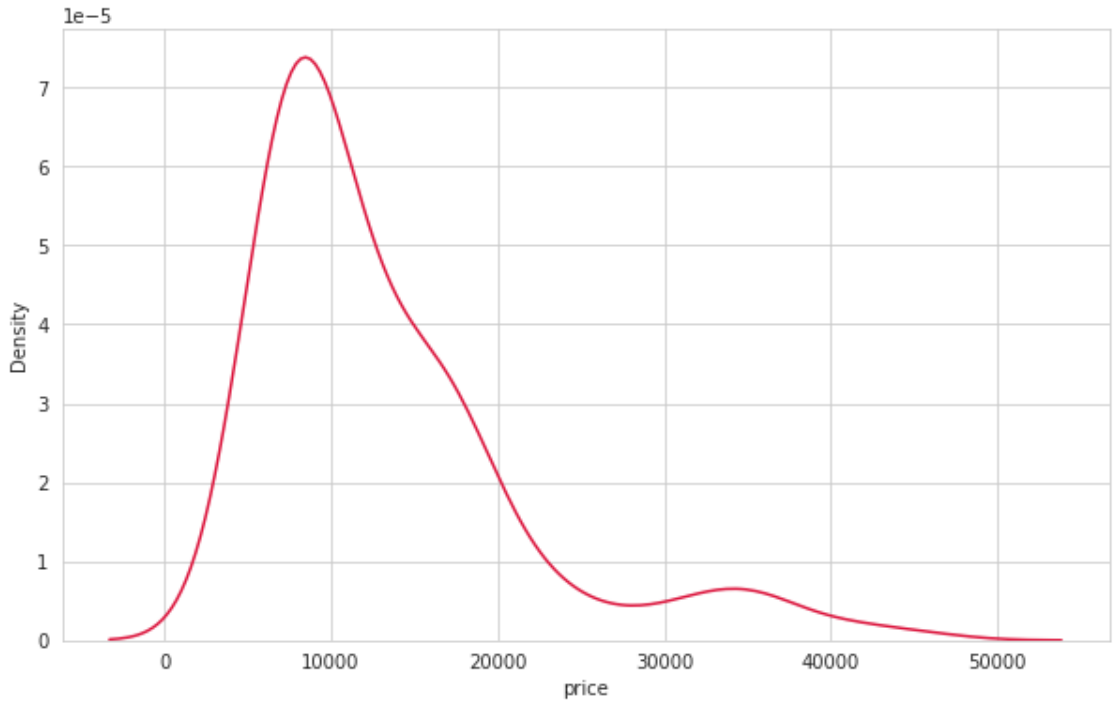


0.0.4 Distribution Plot

- Histogram + Density
- Probability Density Function - Continuous

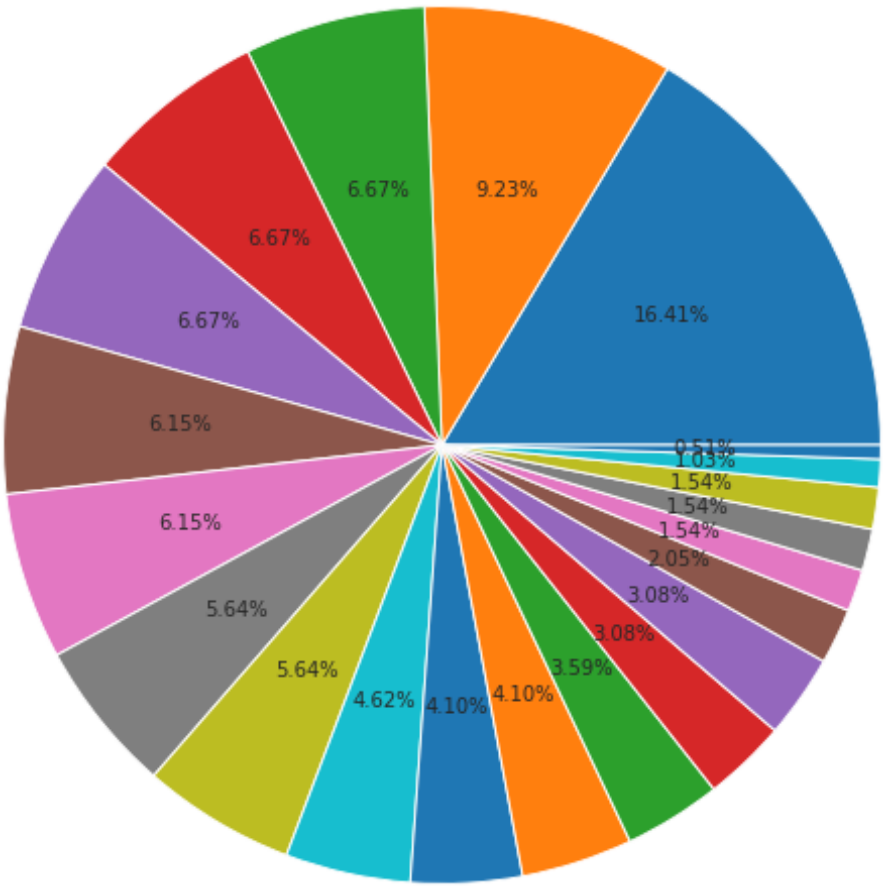
[22]:
```
plt.figure(figsize = (10,6))
# hist = False : Disable Histogram
sns.distplot(auto_price['price'], color = 'crimson', hist = False)
plt.show()
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
```

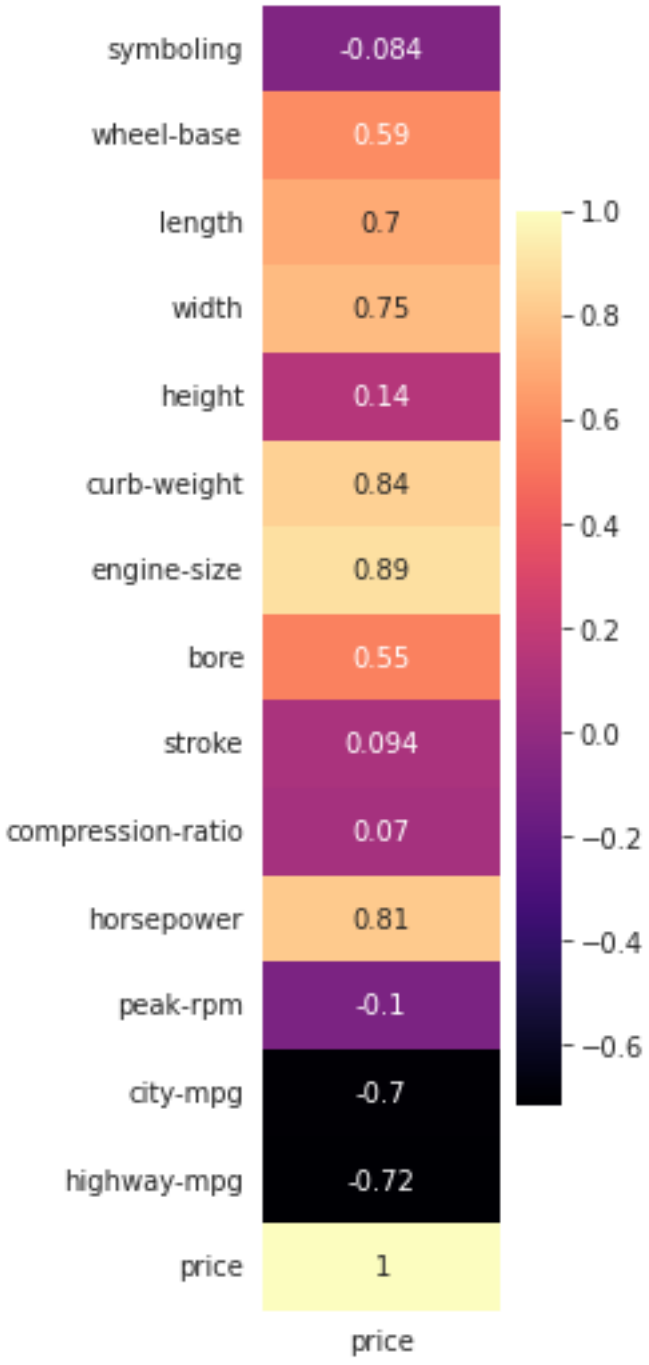


0.0.5 Pie Chart

[24]:
```
plt.figure(figsize = (10,10))
# Shows distribution % in upto 2 decimal Places
plt.pie(auto_price['make'].value_counts(), autopct = '%0.2f%%')
plt.show()
```



[25]:
```
plt.figure(figsize = (2,9))
sns.heatmap(auto_price.corr()[['price']], annot = True, cmap = 'magma')
plt.show()
```


symboling
-0.084
wheel-base
0.59
length
0.7
width
0.75
height
0.14
curb-weight
0.84
engine-size
0.89
bore
0.55
stroke
0.094
compression-ratio
0.07
horsepower
0.81
peak-rpm
-0.1
city-mpg
-0.7
highway-mpg
-0.72
price
1
price
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

1 **if b/w two numerical values corr is +ve it means both are directly proprtional, -ve means both are inversly proprtional.**

[ ]:

[ ]: