Created by Kelvin Tang

Matplotlib and Seaborn

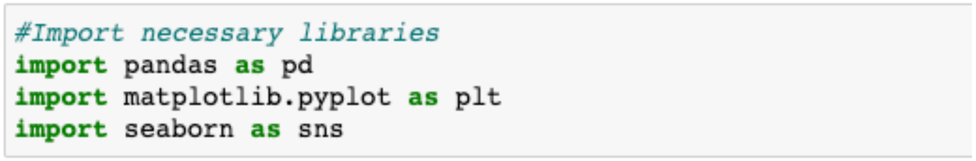
Data visualizing with Python

What is data visualization? Data visualization is a crucial process of transforming a bunch of data into a visual context e.g. charts, graphs. The visualized context will help people easily understand a huge amount of data in one graph. Thus, extract insights about the valuable information hidden behind the data. It can form a solid foundation for the steps later on including data cleaning, Exploratory Data Analysis and Machine Learning models and algorithms.

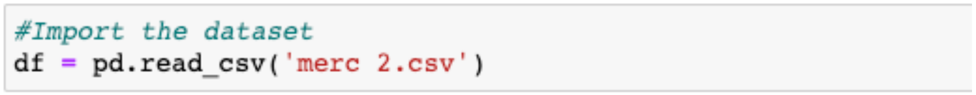
Python offers several great visualizing libraries, such as Pandas, Matplotlib, Seaborn, ggplot and Plotly. In this blog, I will mainly focus on steps I have learned to visualize the data with Matplotlib and Seaborn.

Import libraries and dataset

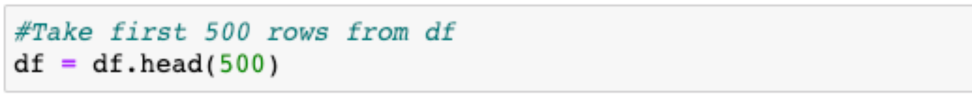
Three libraries are imported including Pandas, Matplotlib and Seaborn. Next step, a CSV file is imported by using pandas 'read_csv' method. Since there are so many rows in the dataset, data-frame is simplified to the first 500 rows only in order to better demonstrate data visualization later on.



```python
#Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```



```python
#Import the dataset
df = pd.read_csv('merc 2.csv')
```

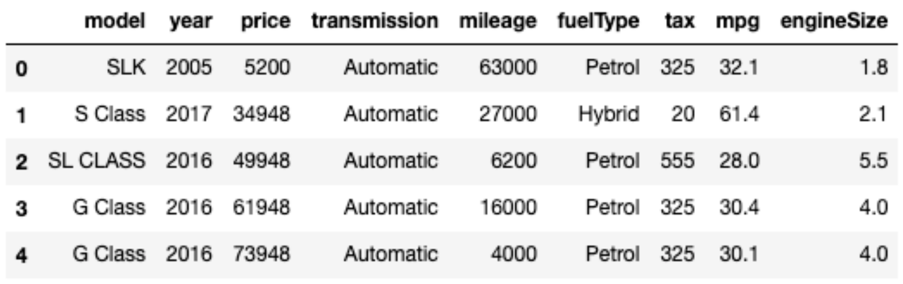


```python
#Take first 500 rows from df
df = df.head(500)
```

Understanding the dataset

This dataset is a simple Mercedes Benz used car listing. It contains 9 columns include:

- Mercedes model
- Registration year
- Price in Euros
- Transmission type
- Mileage: distance used
- fuelType: engine fuel
- tax: road tax
- mpg: miles per gallon
- engineSize: size in liters



	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
0	SLK	2005	5200	Automatic	63000	Petrol	325	32.1	1.8
1	S Class	2017	34948	Automatic	27000	Hybrid	20	61.4	2.1
2	SL CLASS	2016	49948	Automatic	6200	Petrol	555	28.0	5.5
3	G Class	2016	61948	Automatic	16000	Petrol	325	30.4	4.0
4	G Class	2016	73948	Automatic	4000	Petrol	325	30.1	4.0

Matplotlib

Matplotlib is a very popular plotting library for Python programming language. It is an entry-level library with a Matlab like interface. It offers high controllability for programmers to customize wide-range of detail regarding the plot design. Downside-wise, the code is going to be long as more instructions have to be specified which maybe a simple plot.

Seaborn

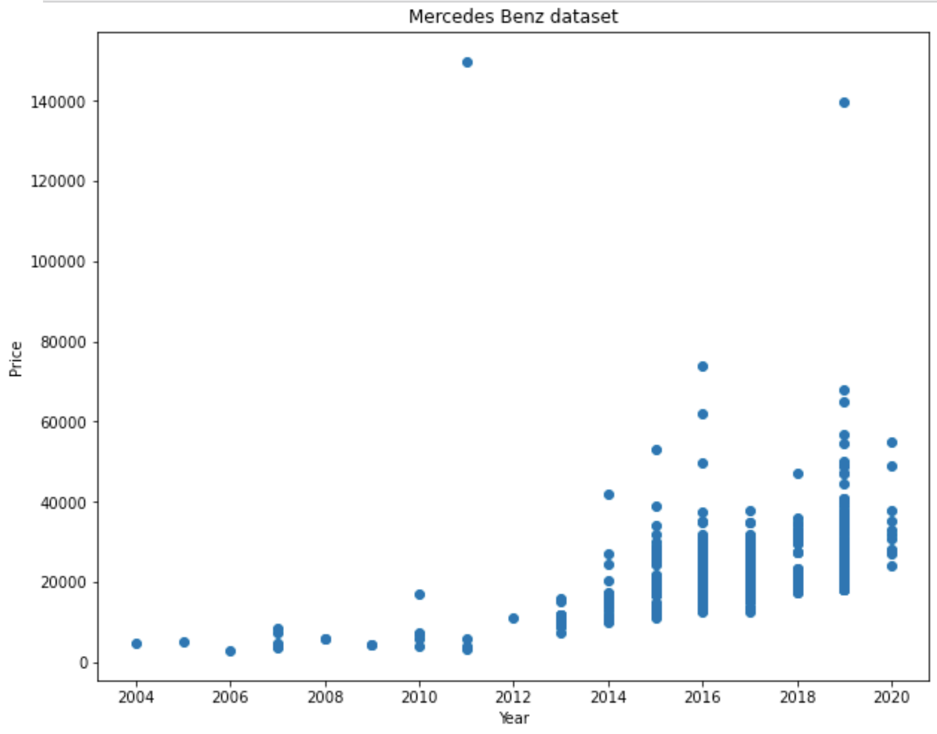
Seaborn is a Python data visualization library based on Matplotlib. It is capable to create sophisticated graphs with a more advanced level interface. Despite its high-level interface, there is just a few lines of codes! Seaborn standard designs generally look very fancy as well.

Scatter plot

-Matplotlib

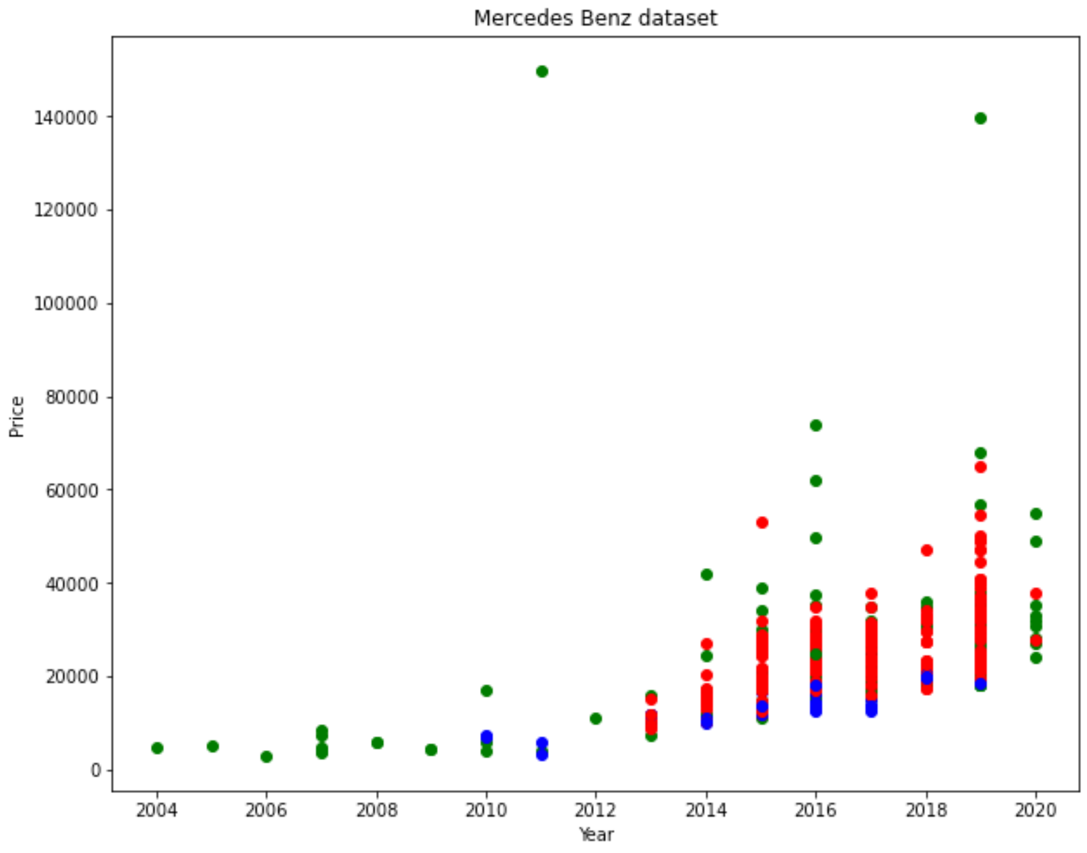
The first graph will be a scatter plot by using matplotlib scatter method, looking at the year and price of the Mercedes. Subplots are created so titles and labels can be added. All figure size in this blog will be set at (10,8). By using ax.scatter, year is on the x-axis and price is on the y-axis. Then we specify the title and x&y labels to visualize the graph clearly. Finally plt.show() is used to plot the graph executing the commands above it.

```
#Scatter plot
fig, ax = plt.subplots(figsize=(10,8))
#year against price
ax.scatter(df['year'], df['price'])
#add title and labels
ax.set_title('Mercedes Benz dataset')
ax.set_xlabel('Year')
ax.set_ylabel('Price')

plt.show()
```



Additionally, we can give different colors to categorize the the transmission of the vehicles. A dictionary is created to map the transmission to color, then use a for-loop assigning the respective color to the scatter points. Eventually, we can obtain a richer content in a single figure as shown below.

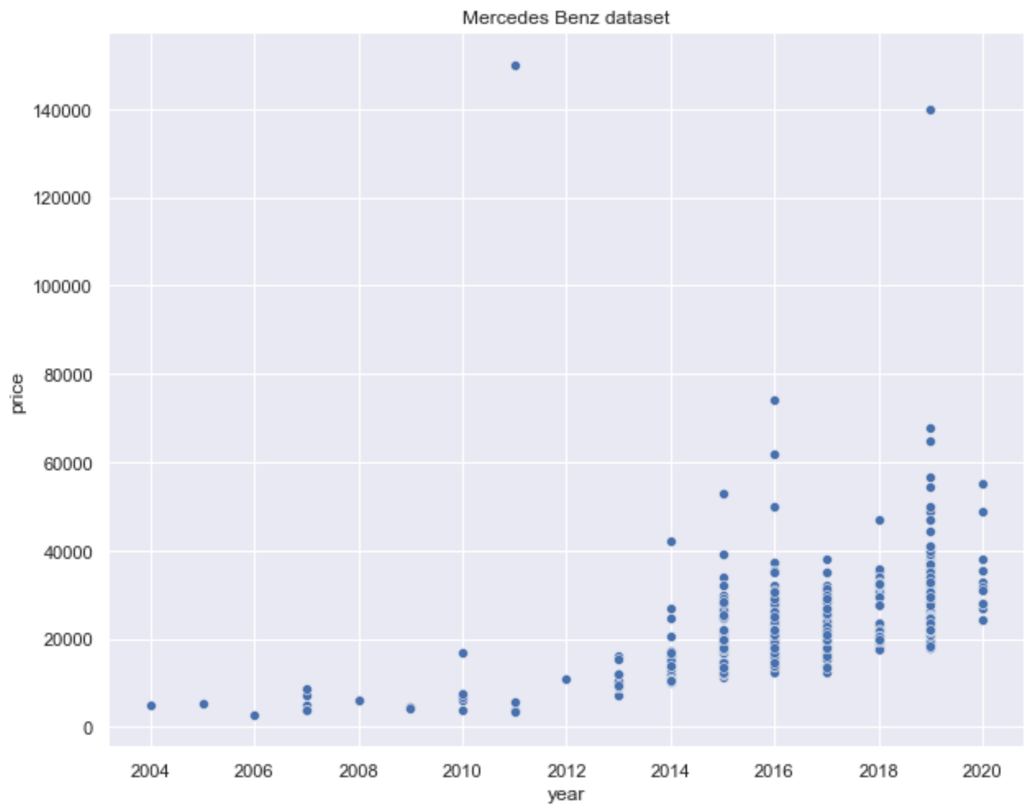
```
#Scatter plot with different colours
colors = {'Semi-Auto':'r', 'Automatic':'g', 'Manual':'b'}

fig,ax = plt.subplots(figsize=(10,8))
for i in range(len(df['year'])):
    ax.scatter(df['year'][i], df['price'][i], color=colors[df['transmission'][i]])
ax.set_title('Mercedes Benz dataset')
ax.set_xlabel('Year')
ax.set_ylabel('Price')
plt.show()
```



-Seaborn

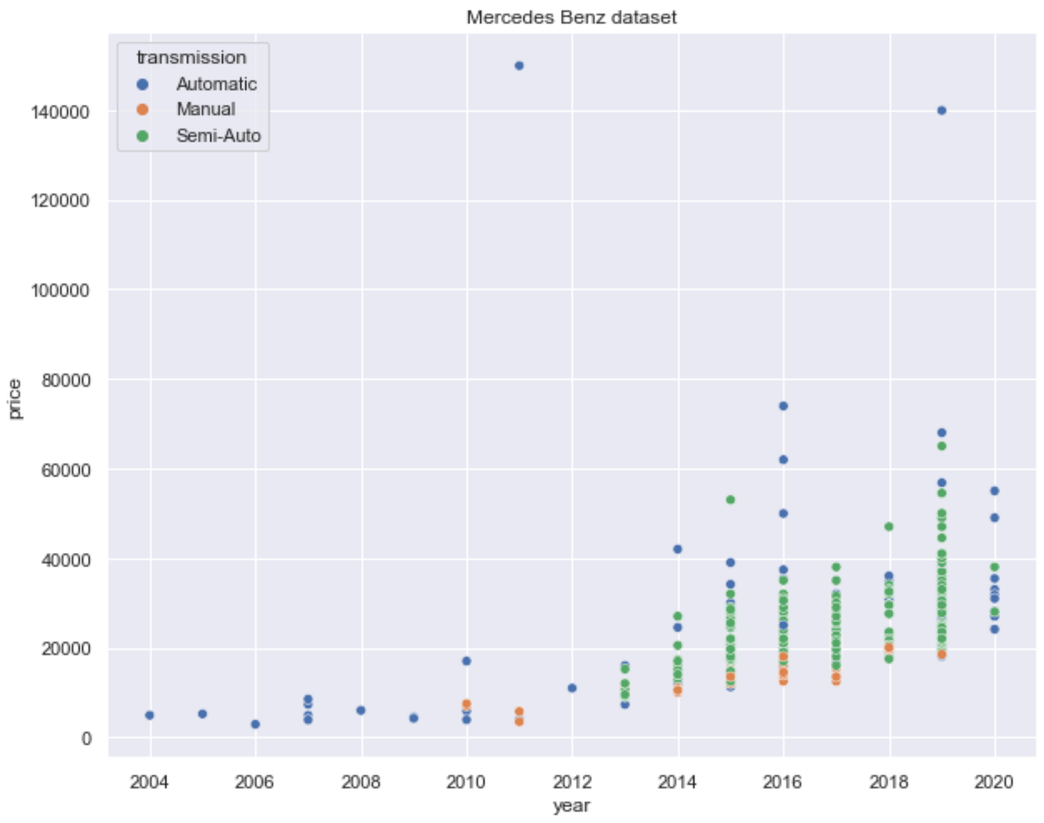
What if we use the same data and plot the graph using seaborn? As mentioned before, Seaborn has simple codes. For this scenario, It will be a single line code! Sns.scatterplot will be used and then insert x-columns, y-columns and data in the bracket. Lastly, set the same title as above. Let’s find out the difference.

```
#Scatter plot
sns.scatterplot(x='year', y='price', data=df).set_title('Mercedes Benz dataset')
```



If we want to add multiple colors, hue argument need to be added.

```
#Scatter plot with colours
sns.scatterplot(x='year', y='price', hue='transmission', data=df).set_title('Mercedes Benz dataset')
```

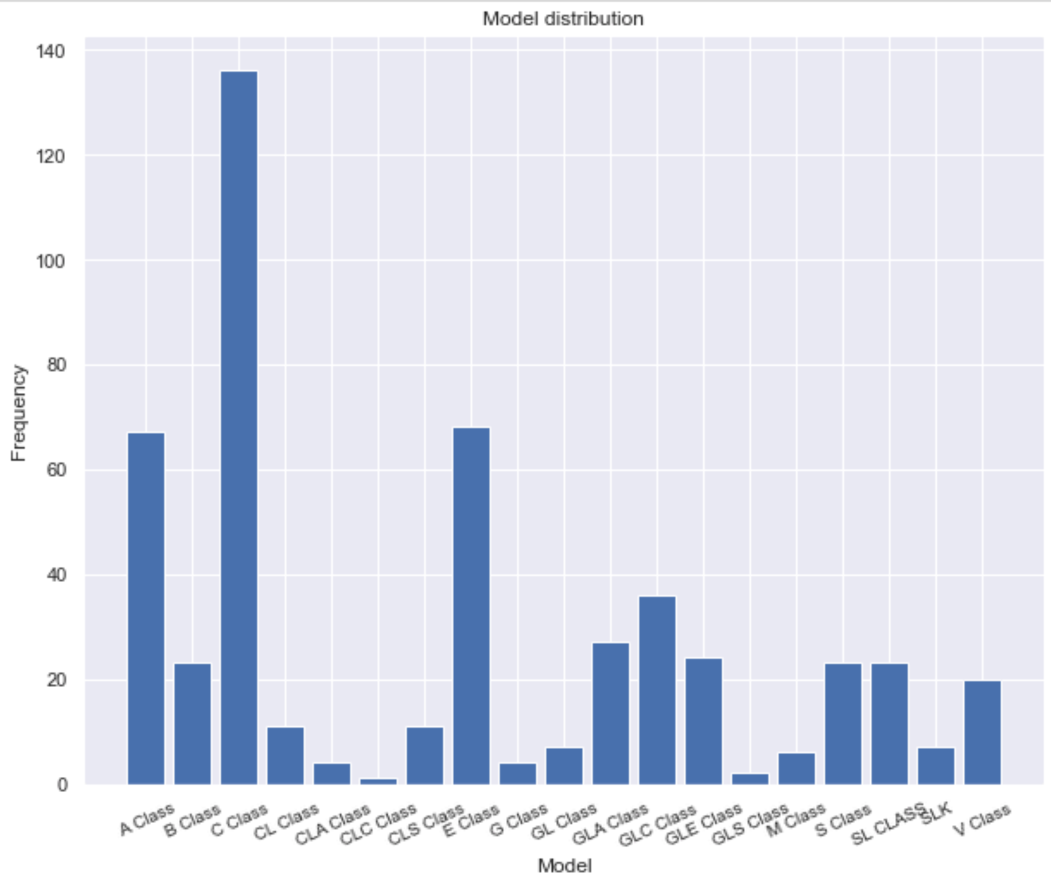


Bar Chart

-Matplotlib

For bar chart, we will look into the Mercedes distribution for each model e.g. A class, B class in the dataset. A bar chart can be created using bar method. Value_counts function has to be added to find out total number of times appeared for each model. Font size and rotation for x-axis labels are amended in order to clarify them without sticking to each other.

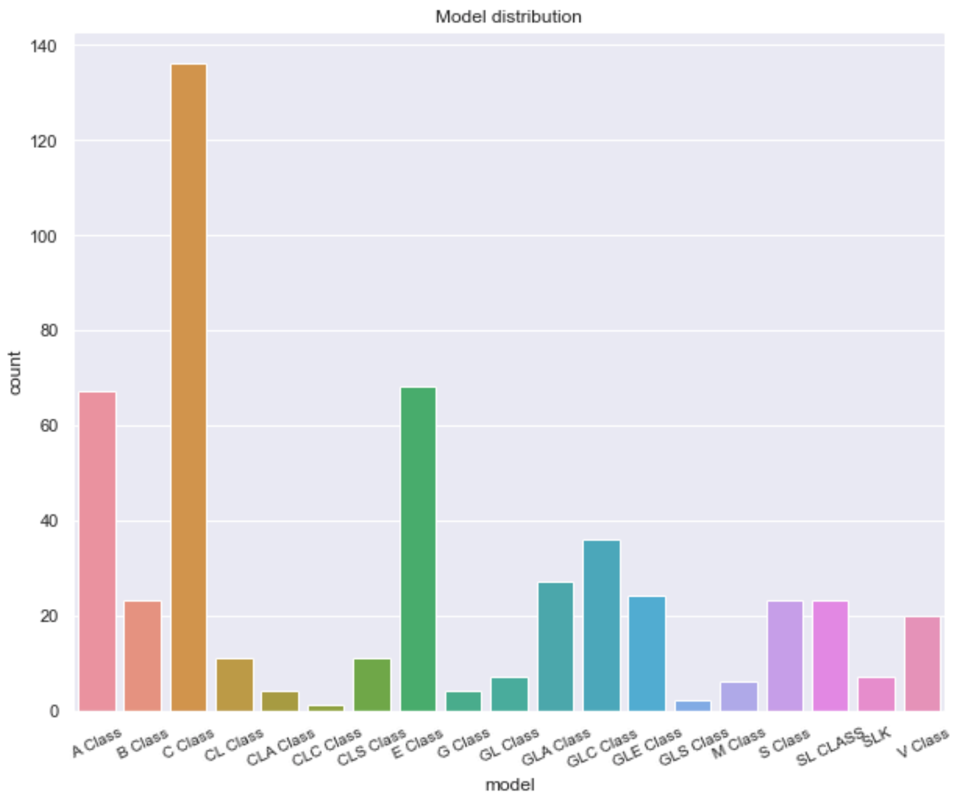
```
#bar chart
fig,ax = plt.subplots(figsize=(10,8))
# count the occurence of each class
data = df['model'].value_counts().sort_index()

#x axis
model = data.index
#y axis
freq = data.values
ax.bar(model,freq)
ax.set_title('Model distribution')
ax.set_xlabel('Model')
ax.set_ylabel('Frequency')
plt.xticks(fontsize=10, rotation=25)
plt.show()
```



-Seaborn

Sns.countplot method can be used to plot a bar chart. It will automatically count the values and various fancy colors for each model. Again, seaborn code is so much simpler.

```
#Bar chart
sns.countplot(x=df['model'].sort_values()).set_title('Model distribution')
plt.xticks(fontsize=10, rotation=25)
plt.show()
```



More graphs with Seaborn

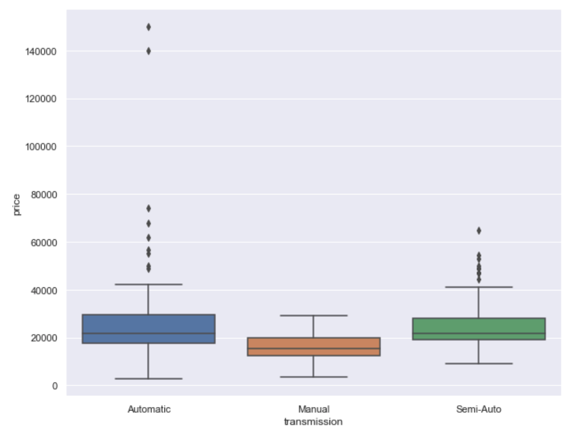
We have found out that Seaborn library is efficient and convenient. So in generally speaking, Seaborn is the to-go-method as beautiful graphs can be plotted in just few lines of codes.

More data will be visualized including box plot, heat map, facet grid and pair plot using Seaborn.

Boxplot

```
#Extra
#boxplot
sns.boxplot(x='transmission', y='price', data=df)
```

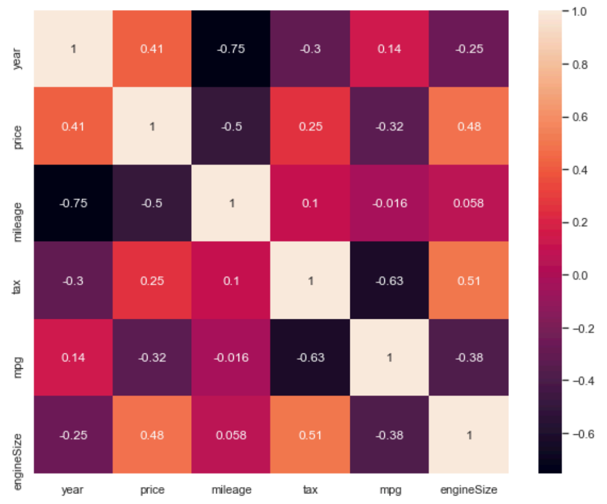
```
<AxesSubplot:xlabel='transmission', ylabel='price'>
```



Heatmap

```
sns.heatmap(df.corr(),annot=True)
```

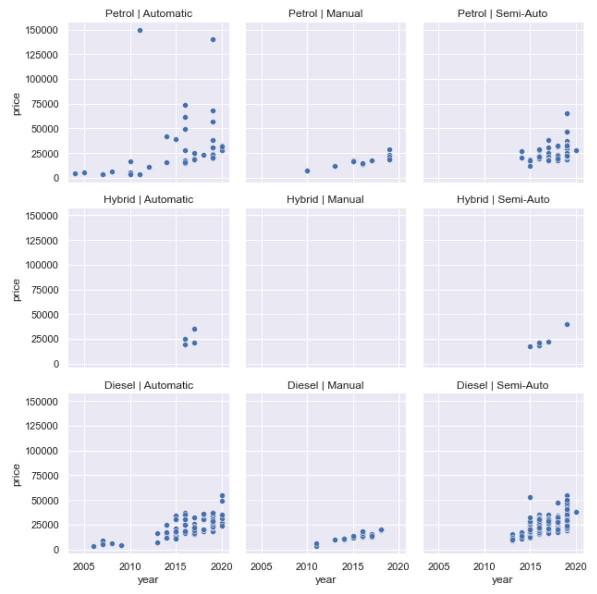
```
<AxesSubplot:>
```



FacetGrid

```
#FacetGrid
g = sns.FacetGrid(df, col='transmission', row='fuelType')
g.map(sns.scatterplot, 'year', 'price')
g.set_titles(row_template = '{row_name}', col_template = '{col_name}')
```

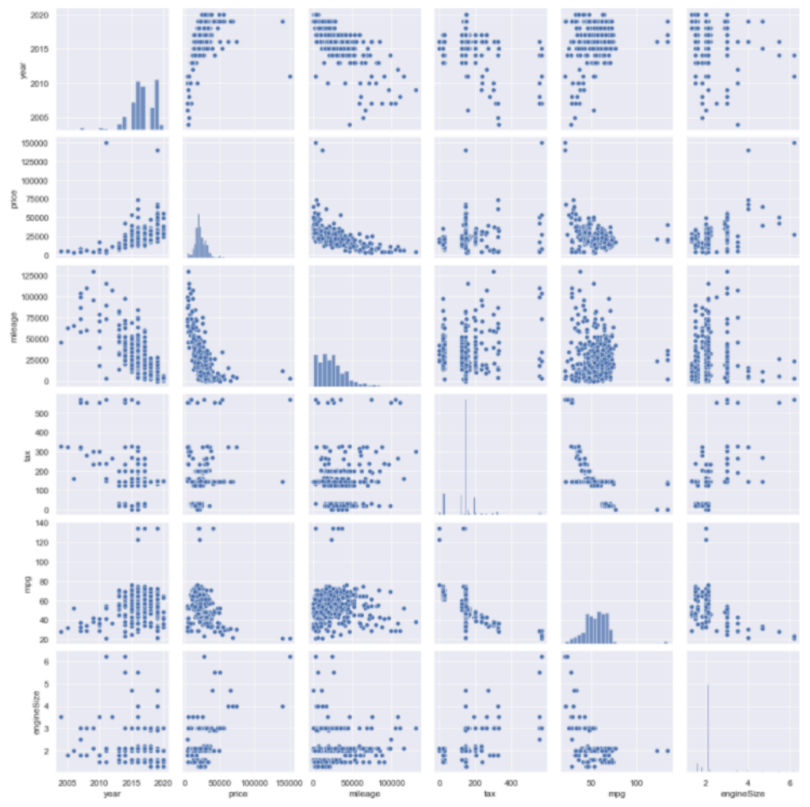
```
<seaborn.axisgrid.FacetGrid at 0x7fbdd855ec70>
```



Pairplot

```
#pairplot
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fbdd855e220>
```



Conclusion

Data visualization is the method to transform data into a visual context, so that human can gain a better understanding of the dataset. Taking this Mercedes case example by using matplotlib and seaborn, data visualizing helped us to understand 500 rows of data, the year and the price by using a single scatter plot. Also, the bar chart clearly showed the number of times appeared for each Mercedes series. Code-wise speaking, seaborn has a much simpler code than matplotlib, it can save programmers a lot of time.

Research has shown that average person responds far better to visual information compared to just plained text, so perhaps this is where data visualization comes in.

Once we know our dataset, next step will be data cleaning, data analysis and build machine learning model if necessary.