Praktikum 6. Exploratory data analysis and Preprocessing (1)

https://www.kaggle.com/code/imoore/intro-to-exploratory-data-analysis-eda-in-python

https://www.kaggle.com/code/ekami66/detailed-exploratory-data-analysis-with-python

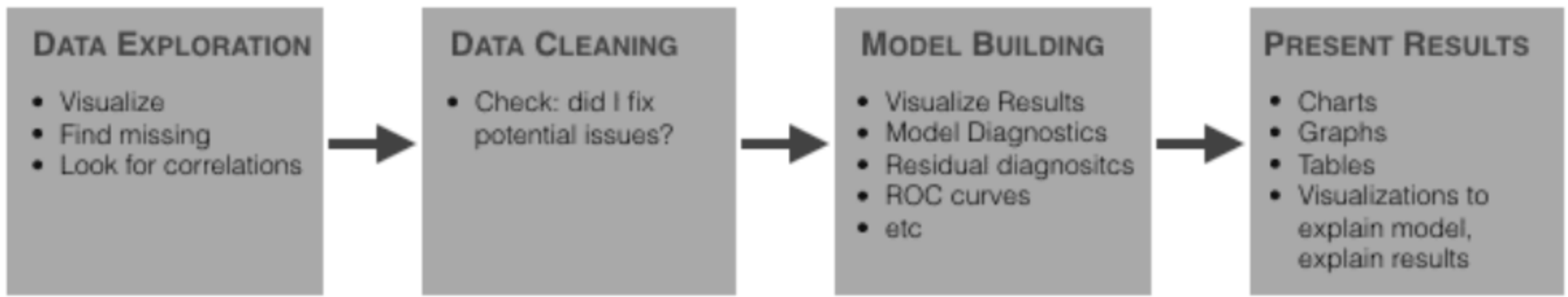
What is Exploratory Data Analysis ?

Analisis Data Eksplorasi atau (EDA) adalah memahami kumpulan data dengan merangkum karakteristik utamanya, sering kali memplotnya secara visual. Langkah ini sangat penting terutama ketika kita sampai pada pemodelan data untuk menerapkan pembelajaran Mesin. Plotting dalam EDA terdiri dari Histogram, Box plot, Scatter plot dan masih banyak lagi. Seringkali dibutuhkan banyak waktu untuk mengeksplorasi data. Melalui proses EDA, kita dapat meminta untuk mendefinisikan pernyataan masalah atau definisi pada kumpulan data kita yang sangat penting

Bagaimana cara melakukan Analisis Data Eksplorasi?

Ini adalah salah satu pertanyaan yang semua orang ingin tahu jawabannya. Jawabannya tergantung pada kumpulan data yang Anda kerjakan. Tidak ada satu metode atau metode umum untuk melakukan EDA, padahal dalam tutorial ini Anda dapat memahami beberapa metode dan plot umum yang akan digunakan dalam proses EDA.

We use data analysis and visualization at every step of the machine learning process



1. Importing the required libraries for EDA

Below are the libraries that are used in order to perform EDA (Exploratory data analysis) in this tutorial.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns                          #visualisation
import matplotlib.pyplot as plt                #visualisation
%matplotlib inline
sns.set(color_codes=True)
```

2. Loading the data into the data frame.

In [2]:

```
df = pd.read_csv("../input/cardataset/data.csv")
# To display the top 5 rows
df.head(5)
```

Out[2]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact

In [3]:

```
df.tail(5)                              # To display the botton 5 rows
```

Out[3]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchb
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchb
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchb
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchb
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0	Luxury

3. Checking the types of data

In [4]:
```
df.dtypes
```

Out[4]:
```
Make                  object
Model                 object
Year                   int64
Engine Fuel Type      object
Engine HP            float64
Engine Cylinders     float64
Transmission Type     object
Driven_Wheels         object
Number of Doors      float64
Market Category       object
Vehicle Size          object
Vehicle Style         object
highway MPG            int64
city mpg               int64
Popularity             int64
MSRP                   int64
dtype: object
```

4. Dropping irrelevant columns

Langkah ini tentunya diperlukan dalam setiap EDA karena terkadang akan banyak kolom yang tidak pernah kita gunakan dalam kasus seperti ini, droping adalah satu-satunya solusi. Dalam hal ini, kolom seperti Jenis Bahan Bakar Mesin, Kategori Pasar, Model Kendaraan, Popularitas, Jumlah Pintu, Ukuran Kendaraan tidak masuk akal bagi saya jadi saya tinggalkan saja untuk contoh ini.

In [5]:
```
df = df.drop(['Engine Fuel Type', 'Market Category', 'Vehicle Style', 'Popularity', 'Number of D
oors', 'Vehicle Size'], axis=1)
df.head(5)
```

Out[5]:

	Make	Model	Year	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	highway MPG	city mpg	MSRP
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

5. Renaming the columns

Dalam hal ini, sebagian besar nama kolom sangat membingungkan untuk dibaca, jadi saya hanya mengubah nama kolomnya. Ini adalah pendekatan yang baik karena meningkatkan keterbacaan kumpulan data.

In [6]:

```
df = df.rename(columns={"Engine HP": "HP", "Engine Cylinders": "Cylinders", "Transmission Type":
"Transmission", "Driven_Wheels": "Drive Mode","highway MPG": "MPG-H", "city mpg": "MPG-C", "MSR
P": "Price" })
df.head(5)
```

Out[6]:

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

6. Dropping the duplicate rows

Hal ini sering kali merupakan hal yang berguna untuk dilakukan karena kumpulan data yang besar seperti dalam kasus ini berisi lebih dari 10.000 baris sering kali memiliki beberapa data duplikat yang mungkin mengganggu, jadi di sini saya menghapus semua nilai duplikat dari kumpulan data. Misalnya sebelum menghapus saya memiliki 11914 baris data tetapi setelah menghapus duplikat 10925 data berarti saya memiliki 989 data duplikat.

In [7]:
```
df.shape
```

Out[7]:
```
(11914, 10)
```

In [8]:
```
duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
```

```
number of duplicate rows:  (989, 10)
```

In [9]:
```
df.count()      # Used to count the number of rows
```

Out[9]:

Make	11914
Model	11914
Year	11914
HP	11845
Cylinders	11884
Transmission	11914
Drive Mode	11914
MPG-H	11914
MPG-C	11914
Price	11914

dtype: int64

Jadi terlihat di atas ada 11914 baris dan saya menghapus 989 baris data duplikat.

In [10]:

```
df = df.drop_duplicates()
df.head(5)
```

Out[10]:

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

In [11]:

```
df.count()
```

Out[11]:

```
Make            10925
Model           10925
Year            10925
HP              10856
Cylinders       10895
Transmission    10925
Drive Mode      10925
MPG-H           10925
MPG-C           10925
Price           10925
dtype: int64
```

7. Dropping the missing or null values.

Ini sebagian besar mirip dengan langkah sebelumnya tetapi di sini semua nilai yang hilang terdeteksi dan kemudian dibuang. Sekarang, ini bukan pendekatan yang baik untuk melakukannya, karena banyak orang hanya mengganti nilai yang hilang dengan mean atau rata-rata kolom tersebut, namun dalam kasus ini, saya hanya menghilangkan nilai yang hilang tersebut. Ini karena ada hampir 100 nilai yang hilang dibandingkan dengan 10.000 nilai. Ini adalah angka yang kecil dan dapat diabaikan jadi saya hilangkan saja nilai tersebut.

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

```
Make              0
Model             0
Year              0
HP               69
Cylinders        30
Transmission      0
Drive Mode        0
MPG-H             0
MPG-C             0
Price             0
dtype: int64
```

In [13]:
```
df = df.dropna()    # Dropping the missing values.
df.count()
```

Out[13]:

```
Make              10827
Model             10827
Year              10827
HP                10827
Cylinders         10827
Transmission      10827
Drive Mode        10827
MPG-H             10827
MPG-C             10827
Price             10827
dtype: int64
```

In [14]:
```
print(df.isnull().sum())    # After dropping the values
```

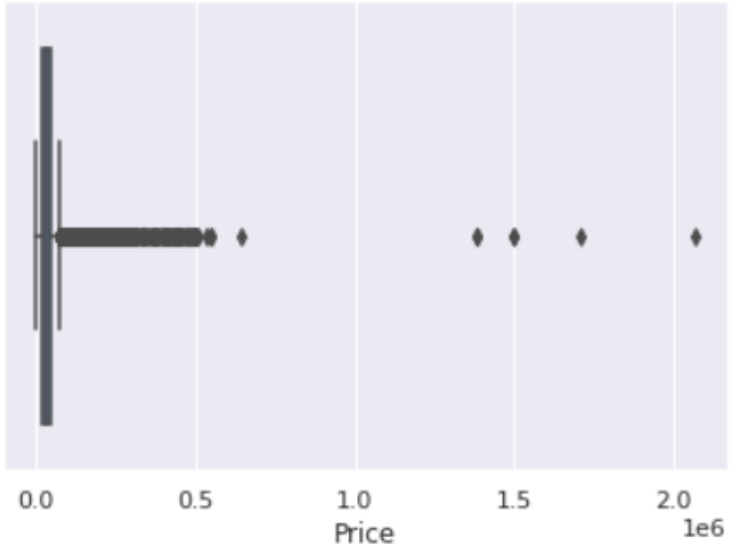
```
Make              0
Model             0
Year              0
HP                0
Cylinders         0
Transmission      0
Drive Mode        0
MPG-H             0
MPG-C             0
Price             0
dtype: int64
```

8. Detecting Outliers

Outlier adalah suatu titik atau kumpulan titik yang berbeda dengan titik lainnya. Terkadang nilainya bisa sangat tinggi atau sangat rendah. Seringkali merupakan ide bagus untuk mendeteksi dan menghilangkan outlier. Karena outlier adalah salah satu alasan utama yang menghasilkan model yang kurang akurat. Oleh karena itu, ada baiknya untuk menghapusnya. Deteksi dan penghapusan outlier yang akan saya lakukan disebut teknik skor IQR. Seringkali outlier dapat dilihat dengan visualisasi menggunakan plot kotak. Di bawah ini adalah plot kotak MSRP, Silinder, Tenaga Kuda, dan Ukuran Mesin. Di sini semua plot, Anda dapat menemukan beberapa titik di luar kotak yang tidak lain adalah outlier. Teknik mencari dan menghilangkan outlier yang saya lakukan pada tugas ini memanfaatkan bantuan tutorial dari ilmu data.

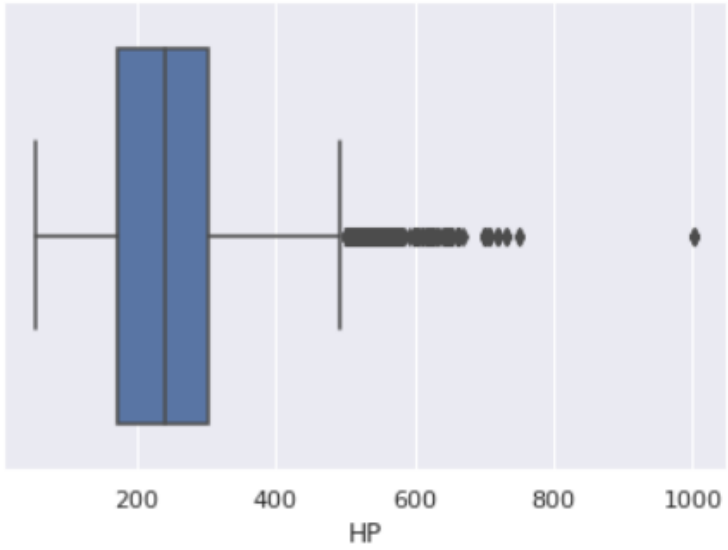
In [15]:
```
sns.boxplot(x=df['Price'])
```

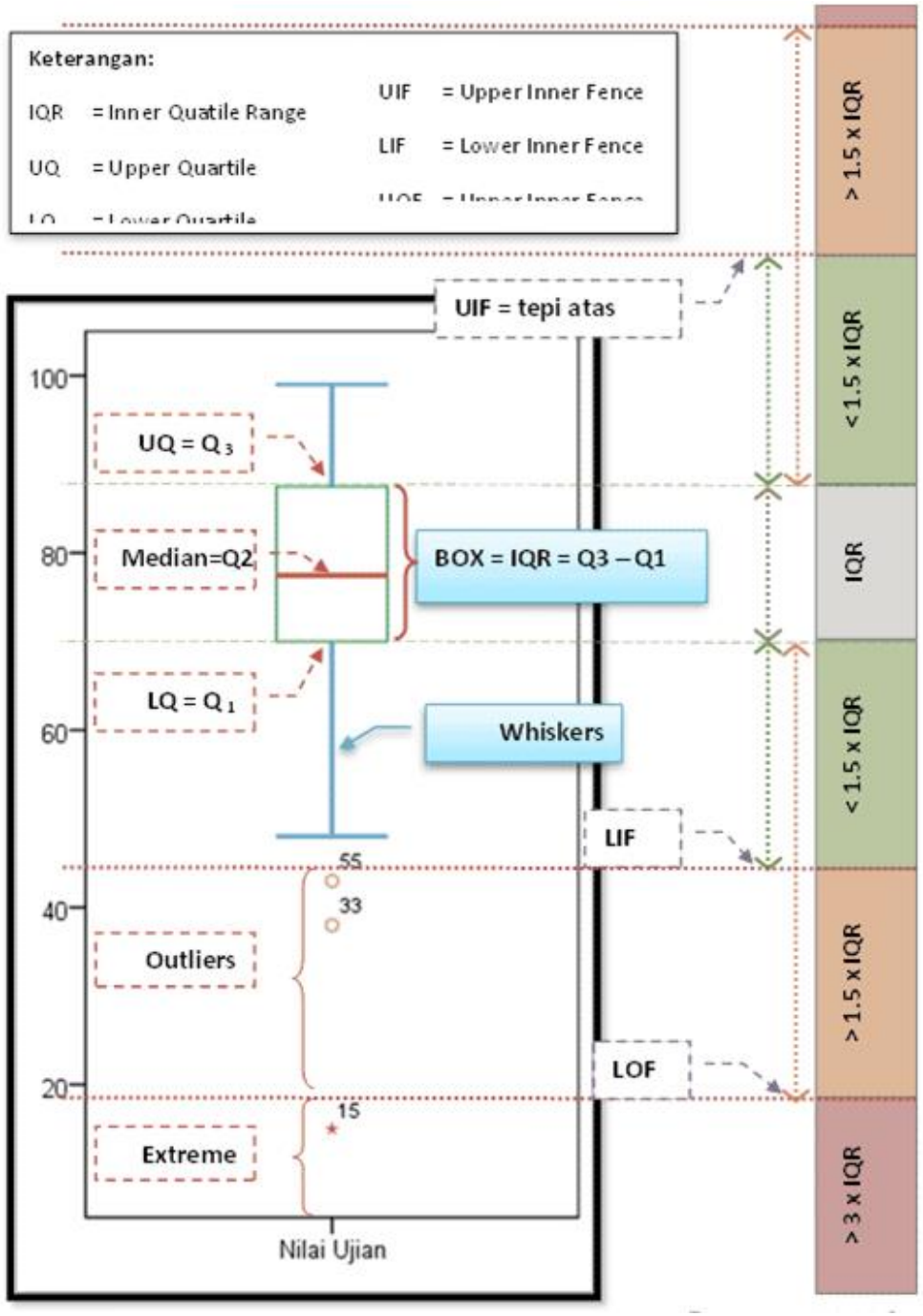
Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89e9c87a90>
```



In [16]:
```
sns.boxplot(x=df['HP'])
```

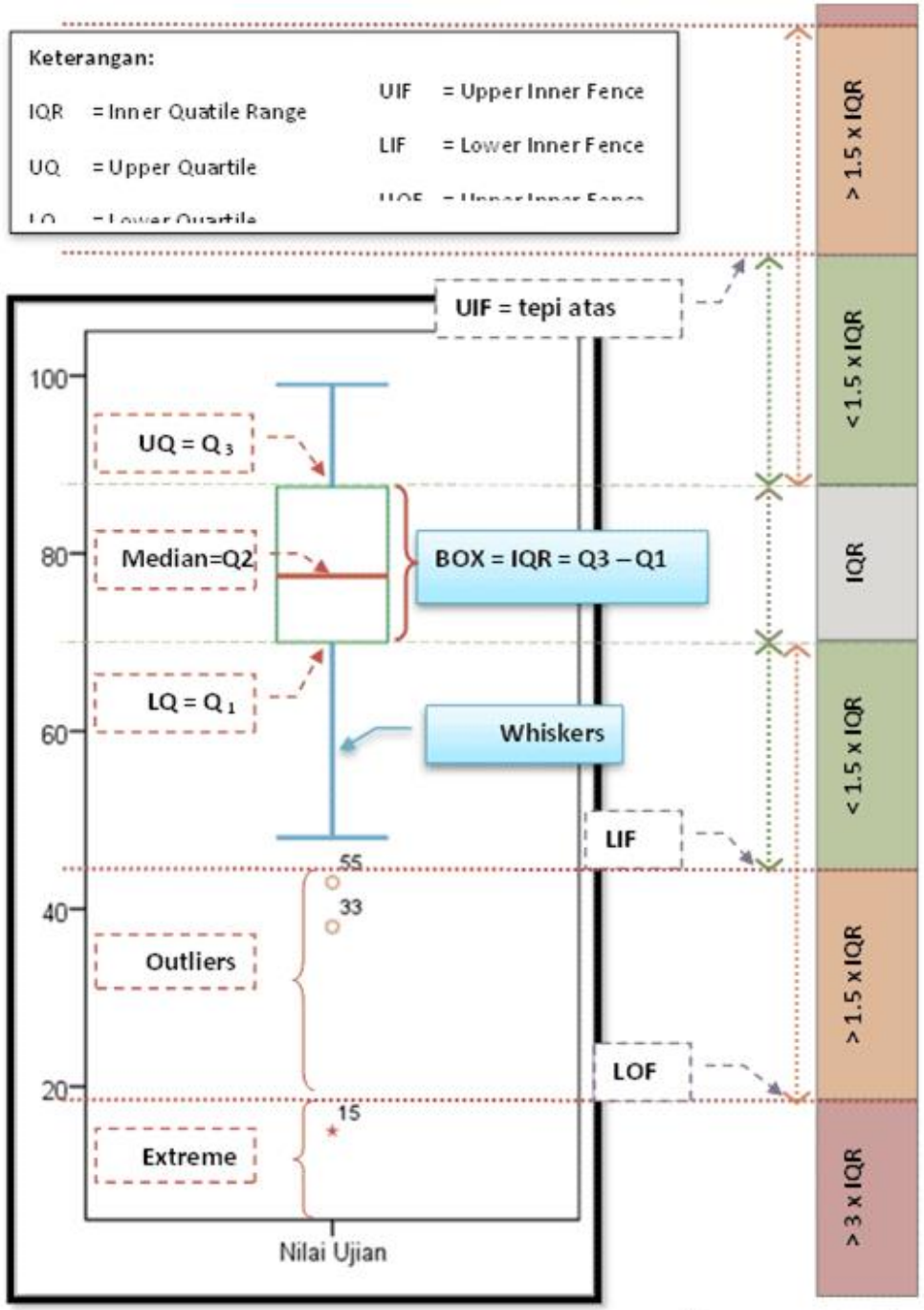
Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89e79e2990>
```





- Bagian utama boxplot adalah kotak berbentuk persegi (**Box**) yang merupakan bidang yang menyajikan interquartile range (**IQR**), dimana 50 % dari nilai data pengamatan terletak di sana.
  - Panjang kotak sesuai dengan jangkauan kuartil dalam (inner Quartile Range, IQR) yang merupakan selisih antara Kuartil ketiga (Q3) dengan Kuartil pertama (Q1). IQR menggambarkan ukuran penyebaran data. Semakin panjang bidang IQR menunjukkan data semakin menyebar. *Pada Gambar, IQR = UQ – LQ* = Q3 – Q1
  - Garis bawah kotak (LQ) = Q1 (Kuartil pertama), dimana 25% data pengamatan lebih kecil atau sama dengan nilai Q1
  - Garis tengah kotak = Q2 (median), dimana 50% data pengamatan lebih kecil atau sama dengan nilai ini
  - Garis atas kotak (**UQ**) = Q3 (Kuartil ketiga) dimana 75% data pengamatan lebih kecil atau sama dengan nilai Q1

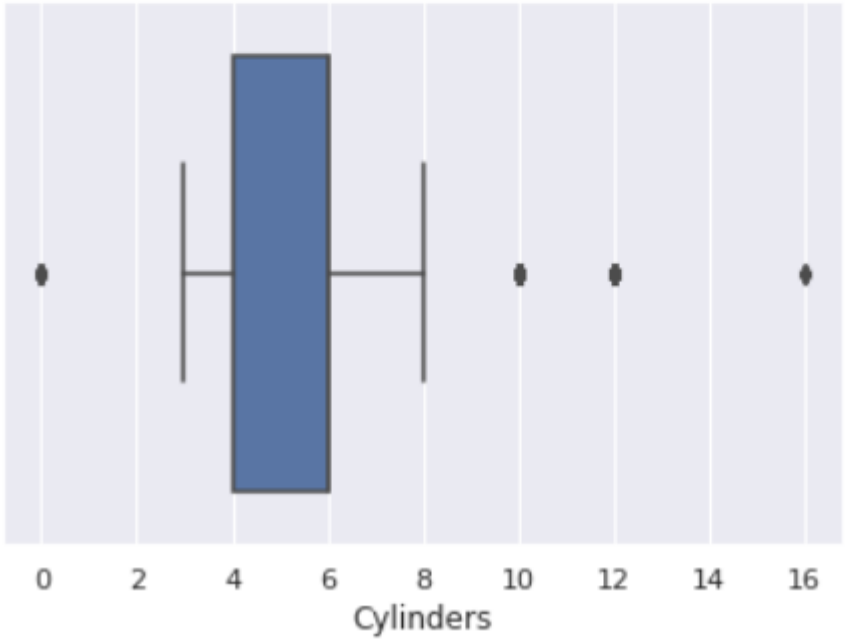
- Garis yang merupakan perpanjangan dari **box**(baik ke arah atas ataupun ke arah bawah) dinamakan dengan **whiskers**.
  - Whiskers bawah menunjukkan nilai yang lebih rendah dari kumpulan data yang berada dalam IQR
  - Whiskers atas menunjukkan nilai yang lebih tinggi dari kumpulan data yang berada dalam IQR
  - Panjang whisker ≤ 1.5 x IQR. Masing-masing garis whisker dimulai dari ujung kotak IQR, dan berakhir pada nilai data yang bukan dikategorikan sebagai outlier (*Pada gambar, batasnya adalah garis UIF dan LIF*). Dengan demikian, nilai terbesar dan terkecil dari data pengamatan (tanpa termasuk outlier) masih merupakan bagian dari Boxplot yang terletak tepat di ujung garis tepi whiskers.



- Nilai yang berada di atas atau dibawah **whisker**dinamakan nilai **outlier** atau **ekstrim**.
  - **Nilai outlier**adalah nilai data yang letaknya lebih dari 1.5 x panjang kotak (IQR), diukur dari UQ (atas kotak) atau LQ (bawah kotak). *Pada Gambar di atas, ada 2 data pengamatan yang merupakan outlier, yaitu data pada case 33 dan case 55 (ada pada baris ke 33 dan baris 35)*
    - Q3 + (1.5 x IQR) < outlier atas ≤ Q3 + (3 x IQR)
    - Q1 – (1.5 x IQR) > outlier bawah ≥ Q1 – (3 x IQR)
  - **Nilai ekstrim**adalah nilai-nilai yang letaknya lebih dari 3 x panjang kotak (IQR), diukur dari UQ (atas kotak) atau LQ (bawah kotak). *Pada gambar di atas, ada 1 data yang merupakan nilai ekstem, yaitu data pada case 15.*
    - Ekstrim bagian atas apabila nilainya berada di atas Q3 + (3 x IQR) dan
    - Ekstrim bagian bawah apabila nilainya lebih rendah dari Q1 – (3 x IQR)

In [17]:
```
sns.boxplot(x=df['Cylinders'])
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89e79e2250>
```



In [18]:
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Year          9.0
HP          130.0
Cylinders     2.0
MPG-H         8.0
MPG-C         6.0
Price     21327.5
dtype: float64
```

Jangan khawatir tentang nilai-nilai di atas karena tidak penting untuk mengetahui masing-masing nilai tersebut karena yang penting adalah mengetahui cara menggunakan teknik ini untuk menghilangkan outlier.

In [19]:
```
df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df.shape
```

Out[19]:
```
(9191, 10)
```

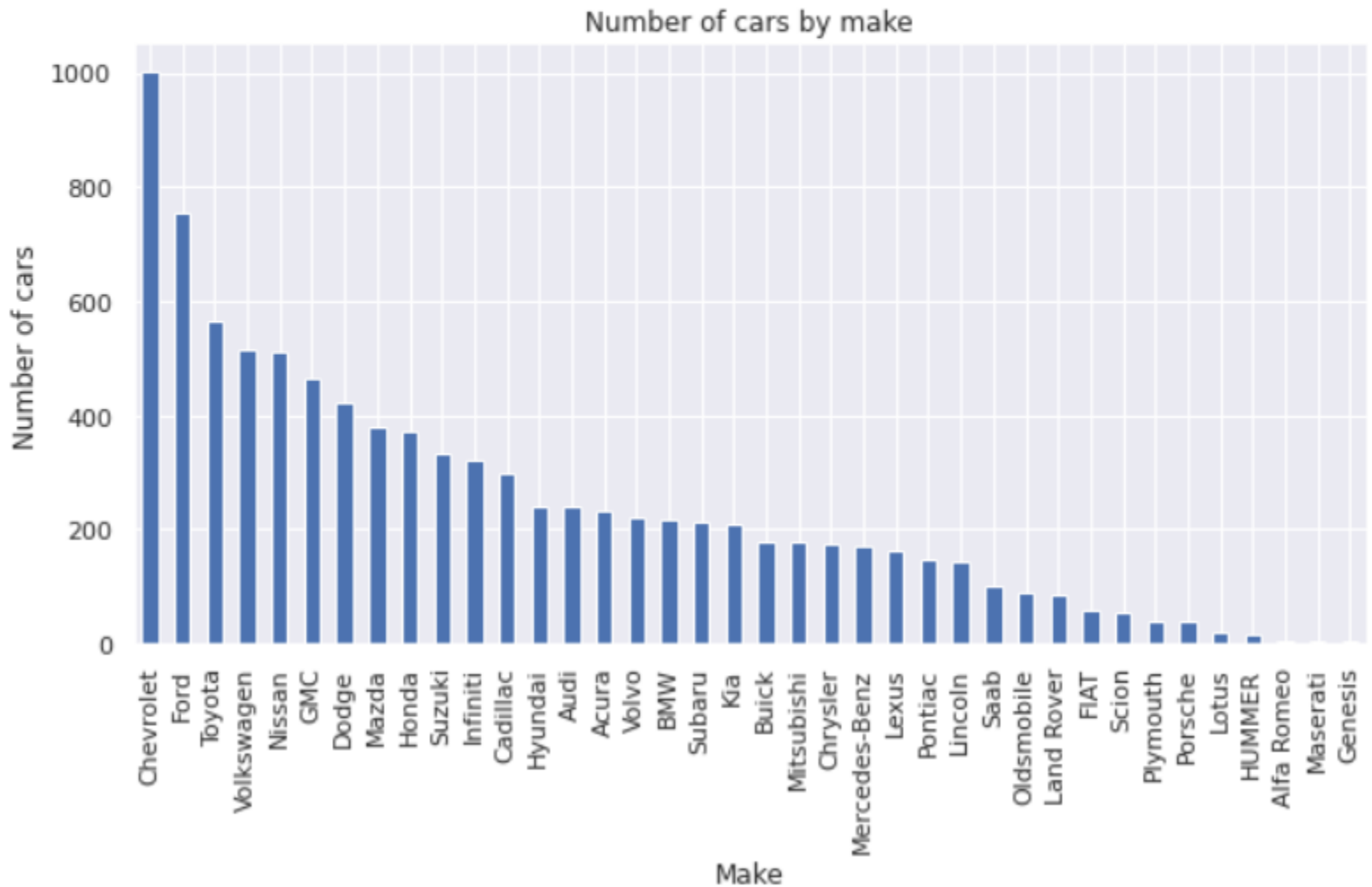
Seperti terlihat di atas, ada sekitar 1.600 baris yang outlier. Namun Anda tidak dapat menghilangkan outlier sepenuhnya karena bahkan setelah Anda menggunakan teknik di atas mungkin masih ada 1–2 outlier yang belum dihilangkan, tetapi tidak masalah karena terdapat lebih dari 100 outlier. Berbuat sesuatu lebih baik daripada tidak sama sekali.

9. Plot different features against one another (scatter), against frequency (histogram)

Histogram
Histogram mengacu pada frekuensi kemunculan variabel dalam suatu interval. Dalam hal ini, terdapat 10 jenis perusahaan manufaktur mobil yang berbeda, namun seringkali penting untuk mengetahui siapa yang memiliki jumlah mobil paling banyak. Melakukan histogram ini adalah salah satu solusi sepele yang memungkinkan kita mengetahui jumlah total mobil yang diproduksi oleh perusahaan berbeda

In [20]:

```
df.Make.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make');
```



Heat Maps

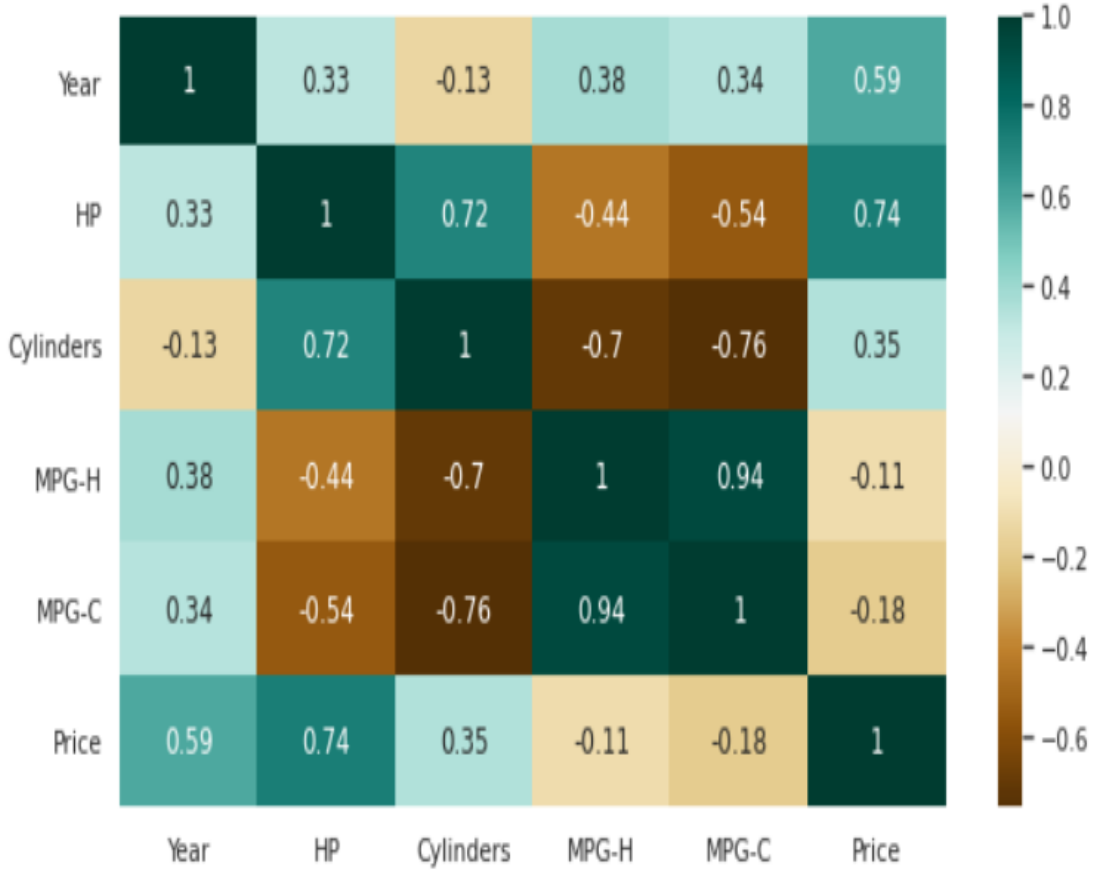
Heat Maps adalah jenis plot yang diperlukan ketika kita perlu mencari variabel terikat. Salah satu cara terbaik untuk menemukan hubungan antar fitur dapat dilakukan dengan menggunakan heat maps. Dalam heat maps di bawah ini kita mengetahui bahwa fitur harga terutama bergantung pada Ukuran Mesin, Tenaga Kuda, dan Silinder.

In [21]:

```
plt.figure(figsize=(10,5))
c= df.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
c
```

Out[21]:

	Year	HP	Cylinders	MPG-H	MPG-C	Price
Year	1.000000	0.326726	-0.133920	0.378479	0.338145	0.592983
HP	0.326726	1.000000	0.715237	-0.443807	-0.544551	0.739042
Cylinders	-0.133920	0.715237	1.000000	-0.703856	-0.755540	0.354013
MPG-H	0.378479	-0.443807	-0.703856	1.000000	0.939141	-0.106320
MPG-C	0.338145	-0.544551	-0.755540	0.939141	1.000000	-0.180515
Price	0.592983	0.739042	0.354013	-0.106320	-0.180515	1.000000



Scatterplot

Saya biasanya menggunakan plot sebar untuk mencari korelasi antara dua variabel. Disini plot sebarnya diplot antara Horsepower dan Price dan kita bisa melihat plotnya di bawah ini. Dengan plot di bawah ini, kita dapat dengan mudah menggambar garis tren. Fitur-fitur ini memberikan penyebaran poin yang baik.

In [22]:

```
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(df['HP'], df['Price'])
ax.set_xlabel('HP')
ax.set_ylabel('Price')
plt.show()
```