

Line Graphs

Line Graph adalah bentuk visualisasi lainya selain diagram lingkaran dan diagram batan. Meskipun diagram lingkaran dan diagram batang berguna untuk menunjukkan bagaimana kelas data saling terkait, diagram garis lebih berguna untuk menunjukkan bagaimana kemajuan data selama beberapa periode. Misalnya, grafik garis dapat berguna dalam membuat grafik suhu dari waktu ke waktu, harga saham dari waktu ke waktu, berat menurut hari, atau metrik berkelanjutan lainnya.

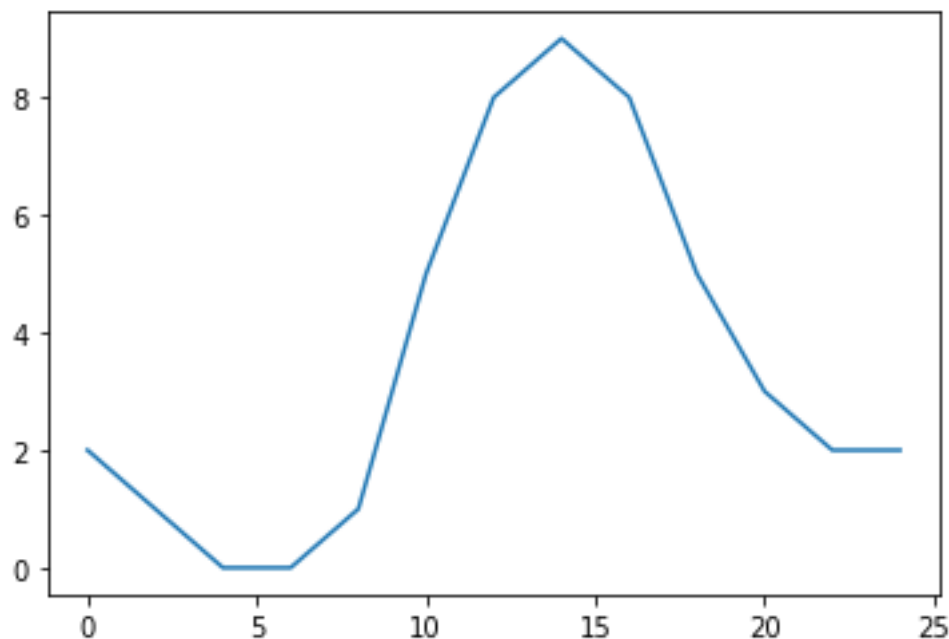
Kita akan membuat grafik garis yang sangat sederhana di bawah ini. Data yang kita miliki adalah suhu dalam celsius dan jam dalam sehari untuk satu hari dan lokasi. Anda dapat melihat bahwa untuk membuat grafik garis kita menggunakan metode `plt.plot()`.

```
import matplotlib.pyplot as plt
```

```
temperature_c = [2, 1, 0, 0, 1, 5, 8, 9, 8, 5, 3, 2, 2]
```

```
hour = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]
```

```
plt.plot(  
    hour,  
    temperature_c  
)  
plt.show()
```



Gambar 0.13. Contoh line graph

Kita dapat melihat bahwa suhu mulai sekitar 2 derajat celcius pada tengah malam, sedikit turun menjadi beku sekitar pukul 05:00, naik menjadi sekitar 9 derajat celcius pada pukul 15:00, dan kemudian turun kembali menjadi sekitar 2 derajat pada tengah malam (Gambar 0.13).

Pertemuan 2

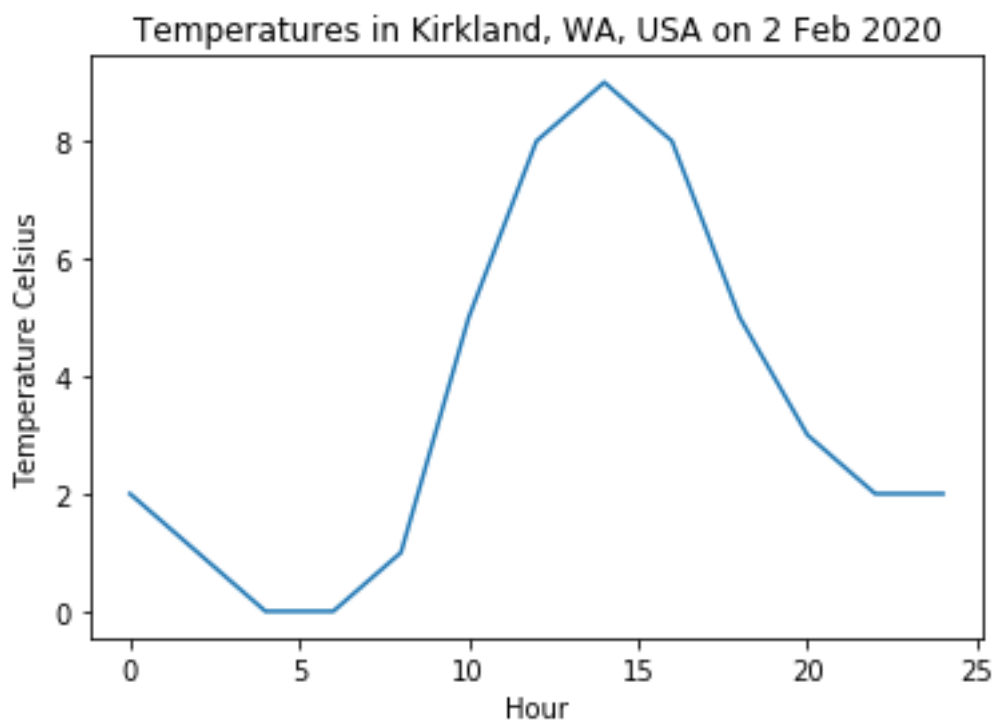
Kita juga bisa menambahkan elemen bagan standar dari title (), ylabel (), dan xlabel () (Gambar 0.14).

```
import matplotlib.pyplot as plt
```

```
temperature_c = [2, 1, 0, 0, 1, 5, 8, 9, 8, 5, 3, 2, 2]
```

```
hour = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]
```

```
plt.plot(  
    hour,  
    temperature_c,  
)  
plt.title('Temperatures in Kirkland, WA, USA on 2 Feb 2020')  
plt.ylabel('Temperature Celsius')  
plt.xlabel('Hour')  
plt.show()
```



Gambar 0.14. Line Graph dengan Title dan Atribut

Kita juga dapat menambahkan penanda di setiap titik data (Gambar 0.15). Pada contoh di bawah ini kita menambahkan penanda titik pada setiap titik data menggunakan argumen `marker = 'o'`.

```
import matplotlib.pyplot as plt
```

```
temperature_c = [2, 1, 0, 0, 1, 5, 8, 9, 8, 5, 3, 2, 2]
```

```
hour = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]
```

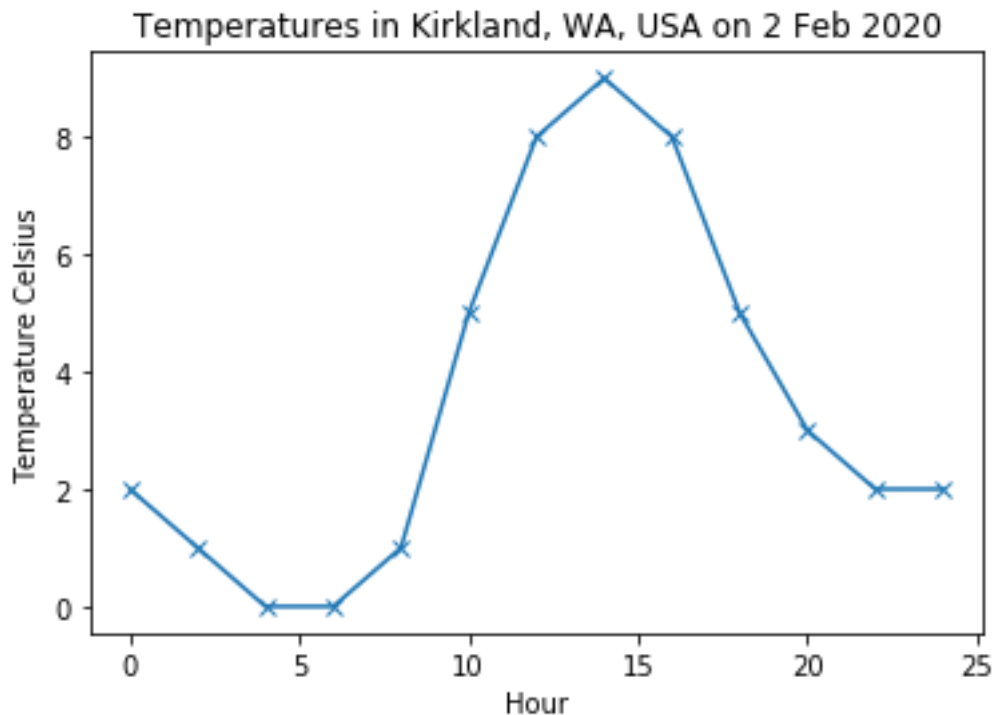
```
plt.plot(  
    hour,  
    temperature_c,  
    marker='o',  
)
```

Pertemuan 2

```

hour,
temperature_c,
marker='x',
)
plt.title('Temperatures in Kirkland, WA, USA on 2 Feb 2020')
plt.ylabel('Temperature Celsius')
plt.xlabel('Hour')
plt.show()

```



Gambar 0.15. Line Graph dengan Penanda disetiap Titik

Kita bahkan dapat memiliki beberapa garis pada grafik yang sama (Gambar 0.16). Misalnya, misalnya, kita ingin mengilustrasikan nilai suhu aktual dan prediksi. Kita bisa memanggil plot () dua kali, sekali dengan setiap kumpulan nilai. Perhatikan bahwa dalam panggilan kedua, kita menggunakan argumen lain untuk plot (), linestyle = '-'. Hal ini menyebabkan garis prediksi terlihat seperti garis putus-putus sedangkan nilai sebenarnya tetap solid.

Anda dapat melihat dokumentasi tambahan pada [Matplotlib pyplot.plot\(\) documentation](#).

```
import matplotlib.pyplot as plt
```

```

temperature_c_actual = [2, 1, 0, 0, 1, 5, 8, 9, 8, 5, 3, 2, 2]
temperature_c_predicted = [2, 2, 1, 0, 1, 3, 7, 8, 8, 6, 4, 3, 3]
hour = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

```

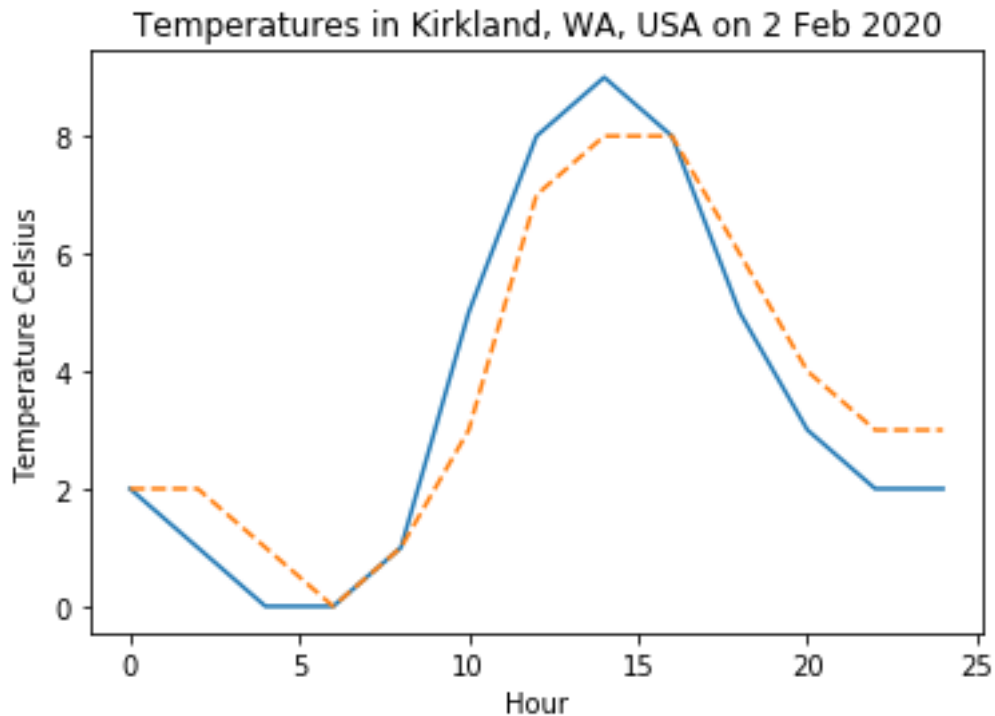
```

plt.plot(hour, temperature_c_actual)
plt.plot(hour, temperature_c_predicted, linestyle='--')

```

Pertemuan 2

```
plt.title('Temperatures in Kirkland, WA, USA on 2 Feb 2020')  
plt.ylabel('Temperature Celsius')  
plt.xlabel('Hour')  
plt.show()
```

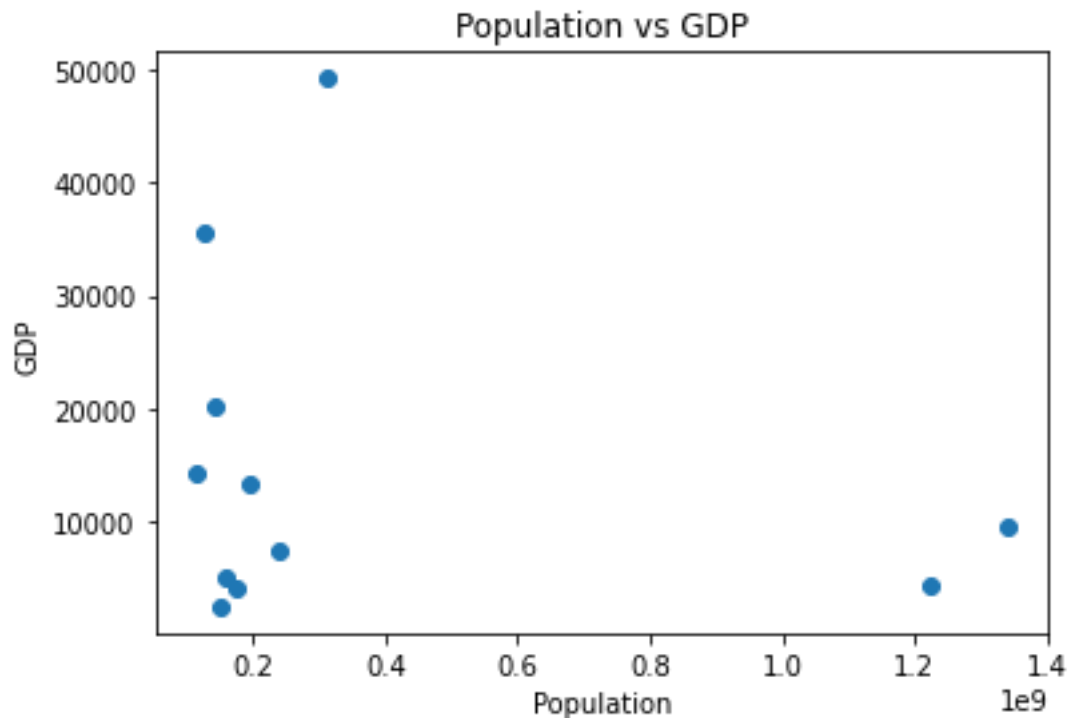


Gambar 0.16. Line Graph dengan Lebih dari satu garis

Scatter Plot

Scatter plot berfungsi baik untuk data dengan dua komponen numerik. Scatter plot dapat memberikan informasi yang berguna terutama mengenai pola atau pencilan. Pada contoh di bawah ini, kita memiliki data yang terkait dengan produk domestik bruto (PDB) dan populasi untuk negara-negara dengan populasi lebih dari seratus juta. PDB adalah total nilai barang dan jasa yang dibuat / disediakan oleh suatu negara selama satu tahun. Kita kemudian menggunakan `plt.scatter ()` untuk membuat sebaran populasi dan PDB (Gambar 0.17).

```
import matplotlib.pyplot as plt
```



Gambar 0.17. Scatter Plot data PDB dan populasi

Scatter plot sangat intuitif karena kita dapat mengumpulkan informasi tentang data kita. Kita dapat melihat bahwa ada dua penciran populasi (penciran PDB). Informasi ini dapat membantu kita memutuskan apakah kita perlu mengoreksi atau mengecualikan penciran dalam analisis kita. Kita juga dapat menambahkan lebih dari satu kumpulan data ke plot (Gambar 0.18).

Pada contoh di bawah ini, kita memplot diameter dan berat sekumpulan lemon dan jeruk nipis agar dapat melihat apakah kita dapat menentukan polanya.

```
import matplotlib.pyplot as plt
```

```
lemon_diameter = [6.44, 6.87, 7.7, 8.85, 8.15, 9.96, 7.21, 10.04, 10.2, 11.06]
```

```
lemon_weight = [112.05, 114.58, 116.71, 117.4, 128.93, 132.93, 138.92, 145.98, 148.44, 152.81]
```

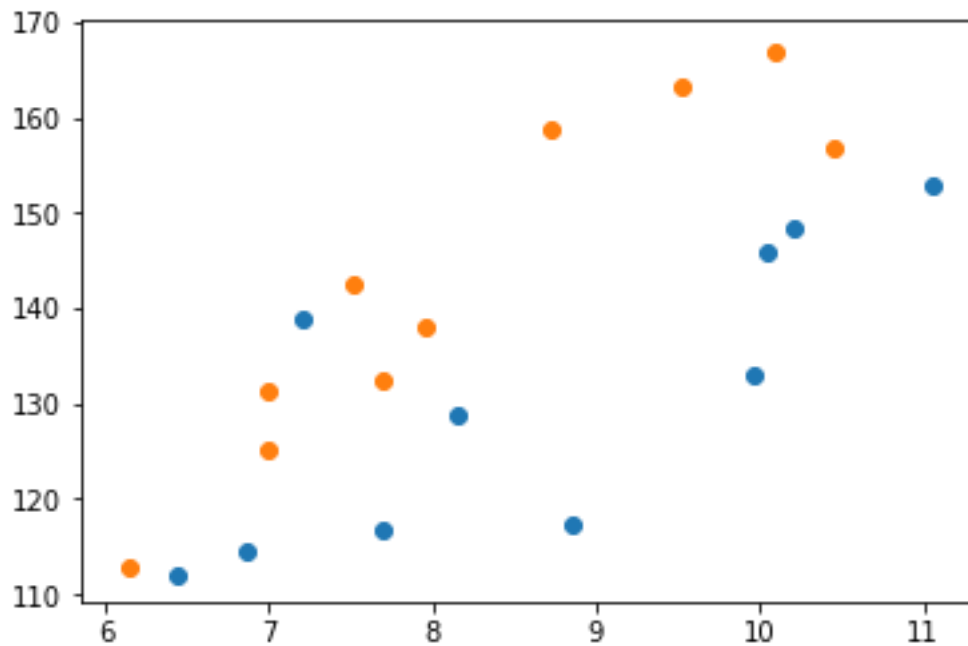
```
lime_diameter = [6.15, 7.0, 7.0, 7.69, 7.95, 7.51, 10.46, 8.72, 9.53, 10.09]
```

```
lime_weight = [112.76, 125.16, 131.36, 132.41, 138.08, 142.55, 156.86, 158.67, 163.28, 166.74]
```

```
plt.scatter(lemon_diameter, lemon_weight)
```

```
plt.scatter(lime_diameter, lime_weight)
```

```
plt.show()
```



Gambar 0.18. Perbandingan lemon dan lime

Melihat sampel kita, tidak ada pola yang sangat jelas. Namun, salah satu jenis jeruk tampaknya lebih berat dan diameternya lebih besar. Tapi yang mana? Mari kita bersihkan bagan ini sedikit. Pertama kita akan menambahkan judul menggunakan `plt.title()`, x-label menggunakan `plt.xlabel()`, dan y-label menggunakan `plt.ylabel()` (Gambar 0.19).

```
import matplotlib.pyplot as plt
```

```
lemon_diameter = [6.44, 6.87, 7.7, 8.85, 8.15, 9.96, 7.21, 10.04,
10.2, 11.06]
```

```
lemon_weight = [112.05, 114.58, 116.71, 117.4, 128.93,
132.93, 138.92, 145.98, 148.44, 152.81]
```

```
lime_diameter = [6.15, 7.0, 7.0, 7.69, 7.95, 7.51, 10.46, 8.72,
9.53, 10.09]
```

```
lime_weight = [112.76, 125.16, 131.36, 132.41, 138.08,
142.55, 156.86, 158.67, 163.28, 166.74]
```

```
plt.title('Lemons vs. Limes')
```

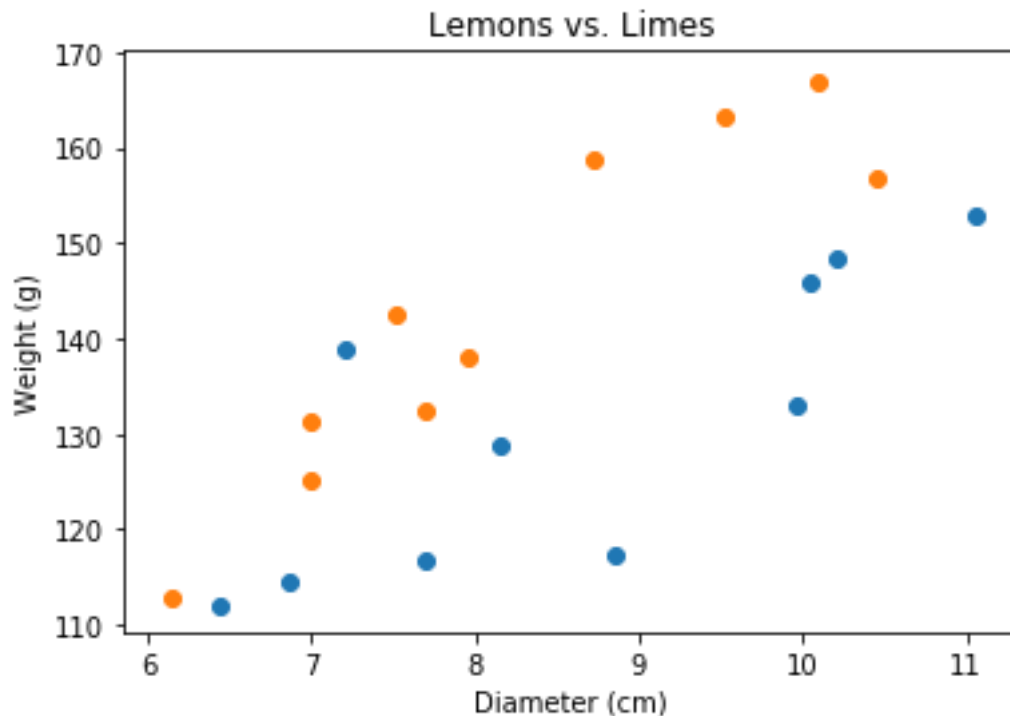
```
plt.xlabel('Diameter (cm)')
```

```
plt.ylabel('Weight (g)')
```

```
plt.scatter(lemon_diameter, lemon_weight)
```

```
plt.scatter(lime_diameter, lime_weight)
```

```
plt.show()
```



Gambar 0.19 Lemon vs lime dengan label

Sekarang kita dapat menambahkan beberapa warna dan legenda untuk membuat scatter plot sedikit lebih intuitif. Kita menambahkan warna dengan meneruskan `color =` argumen ke `plt.scatter()`. Dalam hal ini kita hanya mengatur titik lemon menjadi kuning menggunakan `color = 'y'` dan titik lime menjadi hijau menggunakan `color = 'g'`.

Untuk menambahkan legenda, kita panggil `plt.legend()` dan berikan daftar yang berisi label untuk setiap sebaran data (Gambar 0.20).

```
import matplotlib.pyplot as plt
```

```
lemon_diameter = [6.44, 6.87, 7.7, 8.85, 8.15, 9.96, 7.21, 10.04,
10.2, 11.06]
```

```
lemon_weight = [112.05, 114.58, 116.71, 117.4, 128.93,
132.93, 138.92, 145.98, 148.44, 152.81]
```

```
lime_diameter = [6.15, 7.0, 7.0, 7.69, 7.95, 7.51, 10.46, 8.72,
9.53, 10.09]
```

```
lime_weight = [112.76, 125.16, 131.36, 132.41, 138.08,
142.55, 156.86, 158.67, 163.28, 166.74]
```

```
plt.title('Lemons vs. Limes')
```

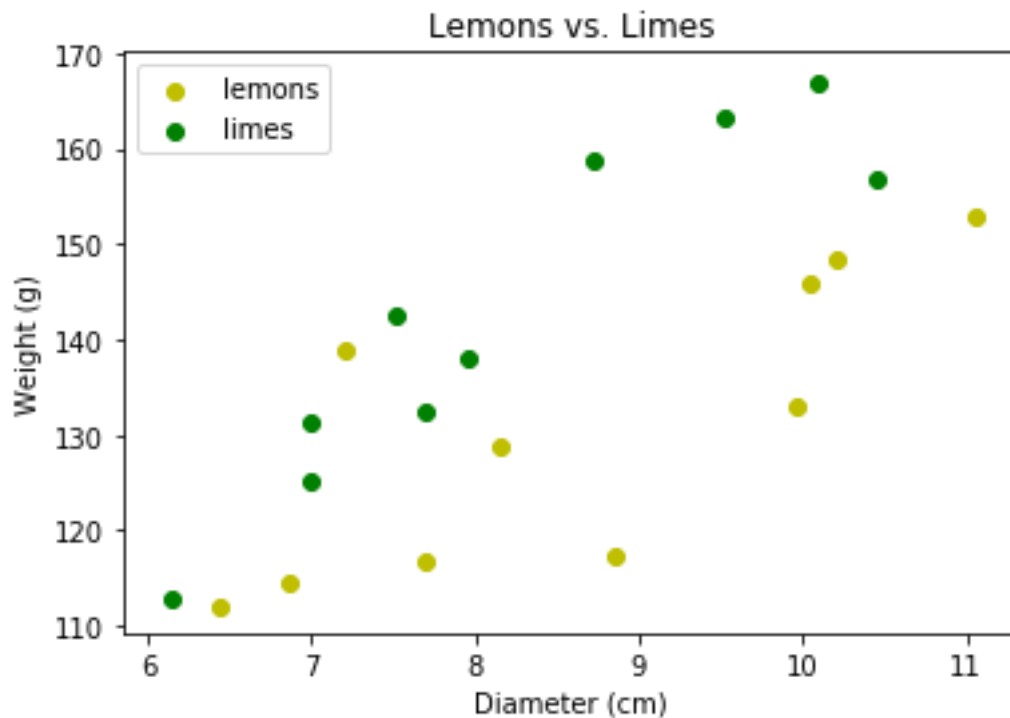
```
plt.xlabel('Diameter (cm)')
```

```
plt.ylabel('Weight (g)')
```

```
plt.scatter(lemon_diameter, lemon_weight, color='y')
```

```
plt.scatter(lime_diameter, lime_weight, color='g')
```

```
plt.legend(['lemons', 'limes'])
plt.show()
```



Gambar 0.20. Lemon vs lime perubahan warna

Sekarang kita dapat melihat lebih jelas bahwa jeruk nipis kita cenderung sedikit lebih berat dan diameternya lebih besar sedikit daripada lemon.

Heatmap

Heatmap adalah jenis visualisasi yang menggunakan kode warna untuk mewakili nilai / kepadatan relatif data di seluruh permukaan. Seringkali ini adalah bagan tabel, tetapi tidak harus terbatas pada itu. Untuk data tabular, terdapat label pada sumbu x dan y. Nilai di persimpangan label tersebut dipetakan ke warna. Warna-warna ini kemudian dapat digunakan untuk memeriksa data secara visual guna menemukan kelompok dengan nilai serupa dan mendeteksi tren dalam data.

Kita akan bekerja dengan data tentang temperatur rata-rata setiap bulan untuk 12 kota terbesar di dunia. Untuk membuat heatmap ini, kita akan menggunakan library Seaborn. Seaborn adalah library visualisasi yang dibangun di atas Matplotlib. Library ini menyediakan antarmuka tingkat yang lebih tinggi dan dapat membuat grafik yang lebih menarik. Langkah-langkah untuk melakukan visualisasi heatmap (Gambar 0.21) adalah sebagai berikut

1. Pada kode di bawah ini, pertama kita mengimpor seaborn.
2. Kita kemudian membuat daftar yang berisi nama 12 kota terbesar di dunia dan 12 bulan dalam setahun.
3. Selanjutnya kita menetapkan daftar-daftar ke variabel suhu. Setiap baris dalam daftar mewakili sebuah kota.

Pertemuan 2

4. Setiap kolom adalah satu bulan. Nilai-nilai tersebut adalah suhu tinggi rata-rata untuk kota selama bulan tersebut.
5. Pada tahap akhir kita memanggil `sns.heatmap()` untuk membuat peta panas. Kita mengirimkan data suhu, nama kota sebagai label-y, dan singkatan bulan sebagai label-x.

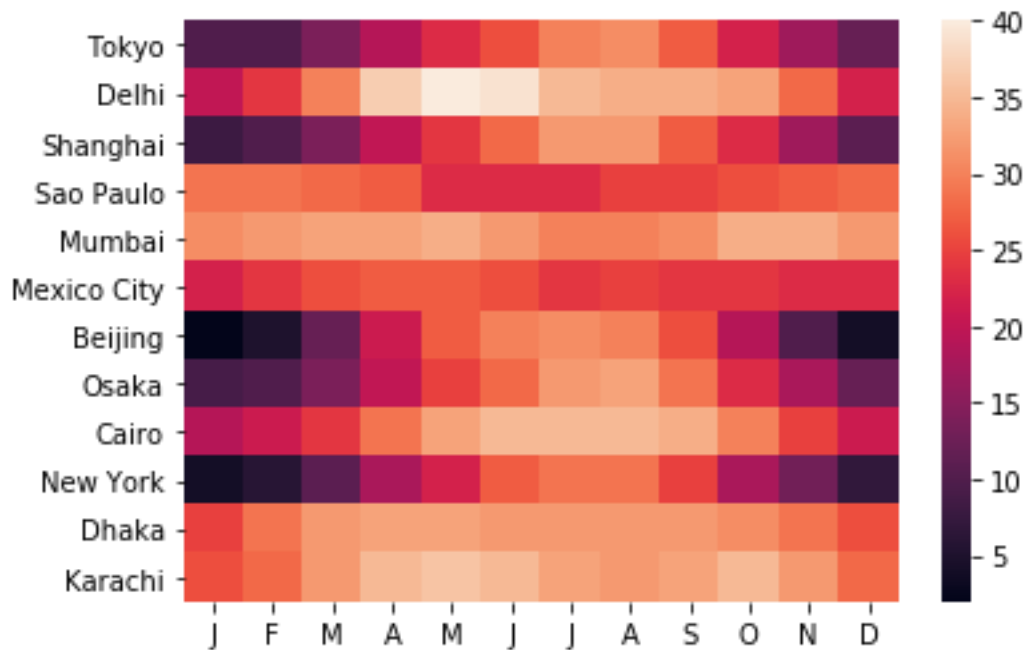
```
import seaborn as sns
```

```
cities = ['Tokyo', 'Delhi', 'Shanghai', 'Sao Paulo', 'Mumbai',  
         'Mexico City',  
         'Beijing', 'Osaka', 'Cairo', 'New York', 'Dhaka',  
         'Karachi']
```

```
months = ['J', 'F', 'M', 'A', 'M', 'J', 'J', 'A', 'S', 'O', 'N',  
         'D']
```

```
temperatures = [  
    [10, 10, 14, 19, 23, 26, 30, 31, 27, 22, 17, 12], # Tokyo  
    [20, 24, 30, 37, 40, 39, 35, 34, 34, 33, 28, 22], # Delhi  
    [ 8, 10, 14, 20, 24, 28, 32, 32, 27, 23, 17, 11], # Shanghai  
    [29, 29, 28, 27, 23, 23, 23, 25, 25, 26, 27, 28], # Sao Paulo  
    [31, 32, 33, 33, 34, 32, 30, 30, 31, 34, 34, 32], # Mumbai  
    [22, 24, 26, 27, 27, 26, 24, 25, 24, 24, 23, 23], # Mexico City  
    [ 2,  5, 12, 21, 27, 30, 31, 30, 26, 19, 10,  4], # Beijing  
    [ 9, 10, 14, 20, 25, 28, 32, 33, 29, 23, 18, 12], # Osaka  
    [19, 21, 24, 29, 33, 35, 35, 35, 34, 30, 25, 21], # Cairo  
    [ 4,  6, 11, 18, 22, 27, 29, 29, 25, 18, 13,  7], # New York  
    [25, 29, 32, 33, 33, 32, 32, 32, 32, 31, 29, 26], # Dhaka  
    [26, 28, 32, 35, 36, 35, 33, 32, 33, 35, 32, 28], # Karachi  
]
```

```
sns.heatmap(temperatures, yticklabels=cities, xticklabels=months)  
<matplotlib.axes._subplots.AxesSubplot at 0x22343c81988>
```



Gambar 0.21. Heatmap mengenai temperatur di masing-masing kota selama 12 bulan

Kita bisa melihat data di grafik yang dihasilkan. Tapi bagaimana kita menafsirkannya? Sebenarnya cukup sulit untuk memahami data. Bagian kiri dan kanan grafik mungkin berisi warna yang lebih gelap, yang memetakan ke suhu yang lebih dingin, tetapi itu pun sulit untuk ditentukan. Kita perlu mengurutkan secara manual kota-kota tersebut, dari yang terkecil ke yang terbesar. Mari kita coba ubah pengurutan berdasarkan lintang pada garis bumi (Gambar 0.22).

```
import seaborn as sns
```

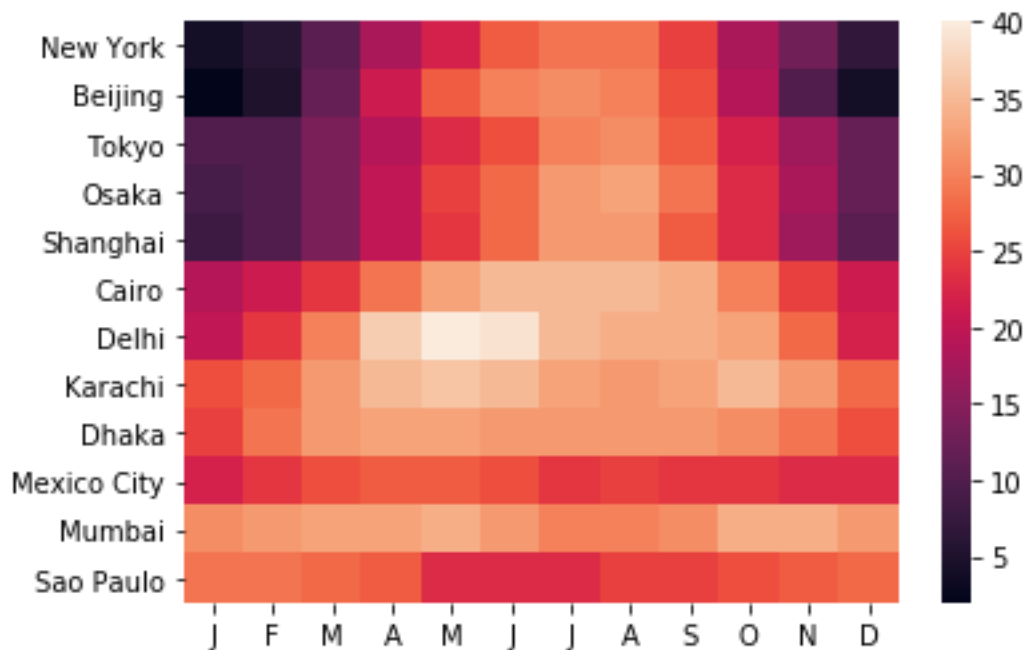
```
cities = ['New York', 'Beijing', 'Tokyo', 'Osaka', 'Shanghai',
          'Cairo', 'Delhi',
          'Karachi', 'Dhaka', 'Mexico City', 'Mumbai', 'Sao Paulo']
```

```
temperatures = [
    [ 4,  6, 11, 18, 22, 27, 29, 29, 25, 18, 13,  7], # New York
    [ 2,  5, 12, 21, 27, 30, 31, 30, 26, 19, 10,  4], # Beijing
    [10, 10, 14, 19, 23, 26, 30, 31, 27, 22, 17, 12], # Tokyo
    [ 9, 10, 14, 20, 25, 28, 32, 33, 29, 23, 18, 12], # Osaka
    [ 8, 10, 14, 20, 24, 28, 32, 32, 27, 23, 17, 11], # Shanghai
    [19, 21, 24, 29, 33, 35, 35, 35, 34, 30, 25, 21], # Cairo
    [20, 24, 30, 37, 40, 39, 35, 34, 34, 33, 28, 22], # Delhi
    [26, 28, 32, 35, 36, 35, 33, 32, 33, 35, 32, 28], # Karachi
    [25, 29, 32, 33, 33, 32, 32, 32, 32, 31, 29, 26], # Dhaka
    [22, 24, 26, 27, 27, 26, 24, 25, 24, 24, 23, 23], # Mexico City
    [31, 32, 33, 33, 34, 32, 30, 30, 31, 34, 34, 32], # Mumbai
```

Pertemuan 2

```
[29, 29, 28, 27, 23, 23, 23, 25, 25, 26, 27, 28], # Sao Paulo
]
```

```
sns.heatmap(temperatures, yticklabels=cities, xticklabels=months)
<matplotlib.axes._subplots.AxesSubplot at 0x22345cc0a48>
```



Gambar 0.22. Heatmap temperature negara berdasarkan garis bumi (lintang)

Kita dapat melihat bahwa kota-kota di garis lintang yang lebih tinggi, lebih dingin dari bulan September hingga Maret dan suhu cenderung meningkat seiring dengan semakin mengecilnya garis lintang.

Perhatikan juga bahwa Sao Paulo terlihat lebih hangat di tengah tahun meskipun berada di belahan bumi selatan. Memang, skema warnanya sulit dibaca. Anda dapat mengubah skema warna menggunakan argumen `cmap =`. `cmap =` menerima daftar warna dan skema warna preset (Gambar 0.23). Anda dapat menemukan skema di [Matplotlib colormap documentation](#).

```
import seaborn as sns
```

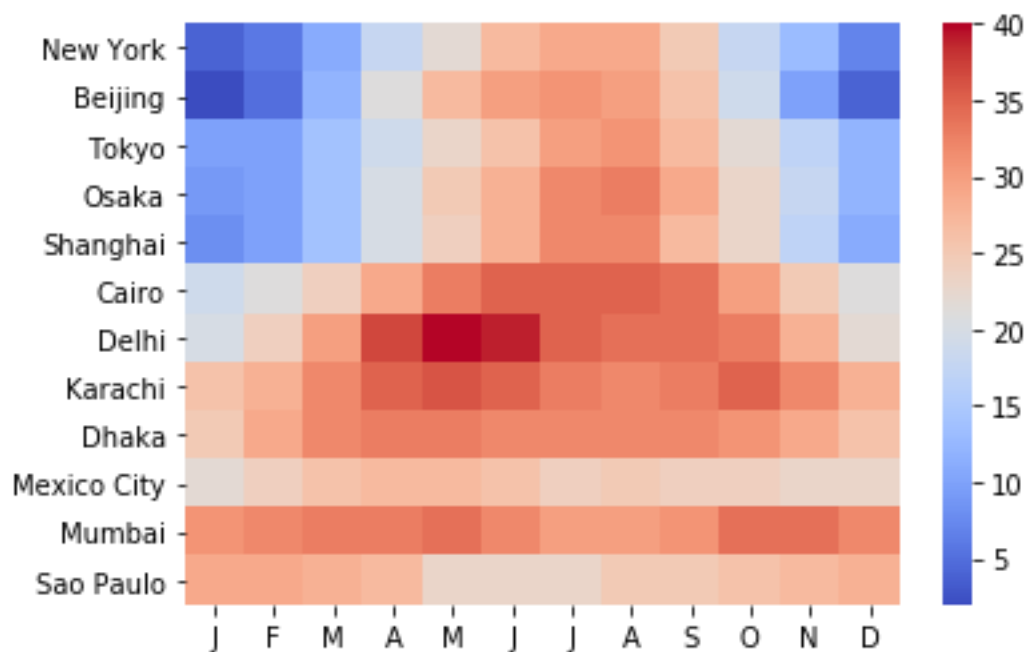
```
cities = ['New York', 'Beijing', 'Tokyo', 'Osaka', 'Shanghai',
          'Cairo', 'Delhi',
          'Karachi', 'Dhaka', 'Mexico City', 'Mumbai', 'Sao Paulo']
```

```
temperatures = [
    [ 4,  6, 11, 18, 22, 27, 29, 29, 25, 18, 13,  7], # New York
    [ 2,  5, 12, 21, 27, 30, 31, 30, 26, 19, 10,  4], # Beijing
    [10, 10, 14, 19, 23, 26, 30, 31, 27, 22, 17, 12], # Tokyo
    [ 9, 10, 14, 20, 25, 28, 32, 33, 29, 23, 18, 12], # Osaka
```

Pertemuan 2

```
[ 8, 10, 14, 20, 24, 28, 32, 32, 27, 23, 17, 11], # Shanghai
[19, 21, 24, 29, 33, 35, 35, 35, 34, 30, 25, 21], # Cairo
[20, 24, 30, 37, 40, 39, 35, 34, 34, 33, 28, 22], # Delhi
[26, 28, 32, 35, 36, 35, 33, 32, 33, 35, 32, 28], # Karachi
[25, 29, 32, 33, 33, 32, 32, 32, 32, 31, 29, 26], # Dhaka
[22, 24, 26, 27, 27, 26, 24, 25, 24, 24, 23, 23], # Mexico City
[31, 32, 33, 33, 34, 32, 30, 30, 31, 34, 34, 32], # Mumbai
[29, 29, 28, 27, 23, 23, 23, 25, 25, 26, 27, 28], # Sao Paulo
]
```

```
sns.heatmap(
    temperatures,
    yticklabels=cities,
    xticklabels=months,
    cmap='coolwarm',
)
<matplotlib.axes._subplots.AxesSubplot at 0x22345d9a3c8>
```



Gambar 0.23. Perubahan warna heatmap colormap

*Visualisasi Statistik***Histogram**

Histogram adalah salah satu visualisasi yang cukup penting dalam memahami distribusi pada data kita. Pandas Histogram menyediakan method yang memudahkan kita untuk membuat histogram. Plot histogram secara tradisional hanya membutuhkan satu dimensi data. Ini dimaksudkan untuk menunjukkan jumlah nilai atau kumpulan nilai secara serial. Pandas `DataFrame.hist()` akan mengambil `DataFrame` kita dan menampilkan plot histogram yang menunjukkan distribusi nilai dalam satu seri. Untuk membuat histogram di panda, yang perlu kita lakukan adalah memberi tahu panda kolom mana yang ingin kita berikan datanya. Dalam hal ini, saya akan memberi tahu panda bahwa saya ingin melihat distribusi harga (histogram).

Parameter lain didalam histogram yang cukup menentukan banyaknya bar didalam visualisasi data kita adalah bin. Cara mudah untuk memikirkan bin adalah "berapa banyak batang yang Anda inginkan dalam bar chart?" Semakin banyak tempat sampah, semakin tinggi resolusi data Anda. Jika kita melakukan setting 2 bin seperti ini tidak terlalu memberikan informasi yang cukup, namun jika kita set menjadi 200 juga terlalu banyak. Kita bisa set sekitar 20-30 agar tampilan seimbang.

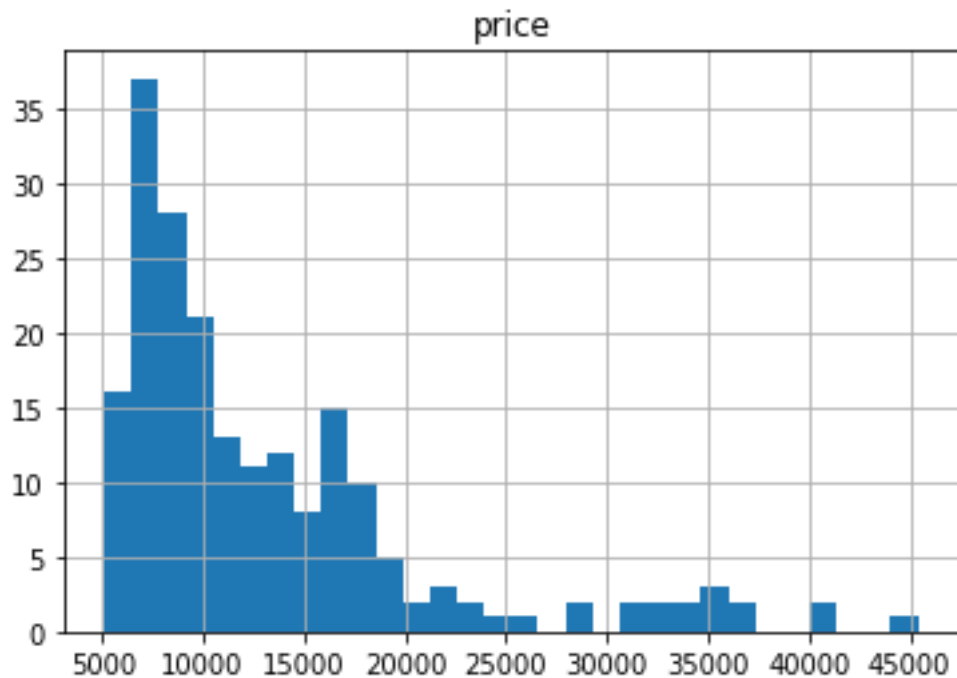
Sebagai contoh studi kasus data yang kita gunakan dalam modul ini adalah `automobile.csv` yang merupakan data-data spesifikasi kendaraan dari berbagai merek dan harganya. Overview data `automobile` dapat dilihat pada Gambar 0.24. Histogram pada data `automobile` dapat kita lihat pada Gambar 0.25, histogram tersebut dibentuk dengan memanggil fungsi `hist()`.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
path='https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-
data/CognitiveClass/DA0101EN/automobileEDA.csv'
df = pd.read_csv(path)
df.head()
```

	symboling	normalized- losses	make	aspiration	num- of- doors	body- style	drive- wheels	engine- location	wheel- base	length	width	height	curb- weight	engine- type	num-of- cylinders	engine- size	fuel- system	bore	stroke	compre
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130	mpfi	3.47	2.68	
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130	mpfi	3.47	2.68	
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	0.909722	52.4	2823	ohcv	six	152	mpfi	2.68	3.47	
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	0.919444	54.3	2337	ohc	four	109	mpfi	3.19	3.40	
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	0.922222	54.3	2824	ohc	five	136	mpfi	3.19	3.40	

Gambar 0.24. Dataset Mobil

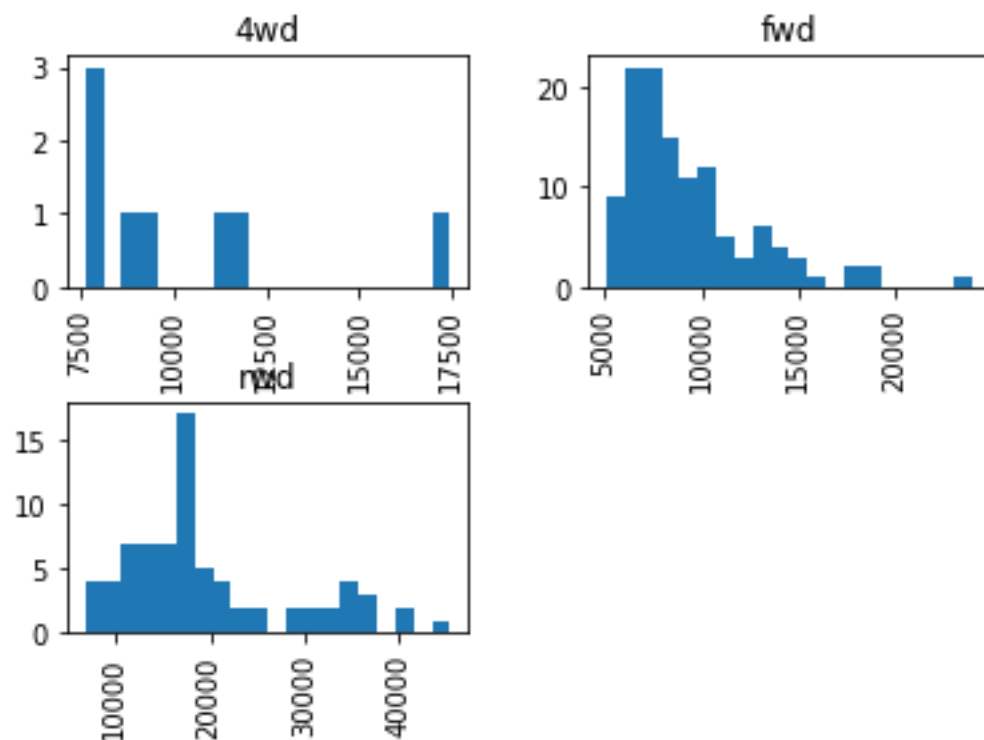
```
df.hist(column='price', bins=30);
```



Gambar 0.25. Contoh Histogram

Kita juga dapat memplot beberapa grup secara berdampingan. Di sini saya ingin melihat dua histogram, histogram price akan dikelompokkan berdasarkan roda penggerak dari kendaraan (fwd – berpenggerak roda depan, 4wd – berpenggerak 4 roda, atau rwd – penggerak belakang (Gambar 0.26).

```
df.hist(column='price', by='drive-wheels', bins=20);
```

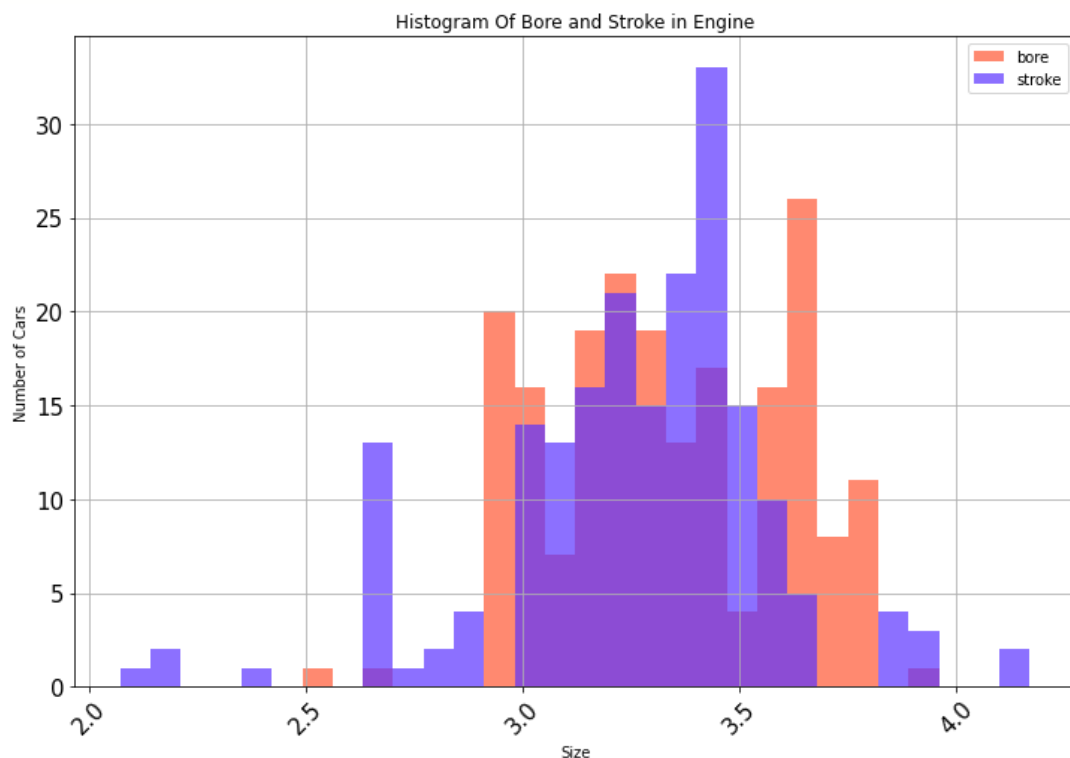


Gambar 0.26. Histogram untuk masing-masing kategori

Pertemuan 3

Untuk memplot beberapa seri, kita bisa menggunakan metode `df.plot(kind='hist')` (Gambar 0.27).

```
df[['bore', 'stroke']].plot(kind='hist',  
    alpha=0.7,  
    bins=30,  
    title='Histogram Of Bore and Stroke in Engine',  
    rot=45,  
    grid=True,  
    figsize=(12,8),  
    fontsize=15,  
    color=['#FF5733', '#5C33FF'])  
plt.xlabel('Size')  
plt.ylabel("Number of Cars");
```



Gambar 0.27. Penggabungan histogram dalam satu visualisasi

Correlation dan Causation

Korelasi merupakan suatu pengukuran sejauh mana nilai saling ketergantungan antar variabel. Causation merupakan hubungan antara sebab dan akibat antara dua variable. Penting untuk mengetahui perbedaan antara keduanya dan bahwa korelasi tidak mendeskripsikan sebab-akibat. Menentukan korelasi jauh lebih sederhana menentukan sebab memerlukan analisis lebih lanjut

Korelasi Pearson

Korelasi Pearson mengukur ketergantungan linier antara dua variabel X dan Y. Koefisien yang dihasilkan adalah nilai antara -1 dan 1 inklusif, di mana:

- 1: Total korelasi linier positif.
- 0 : Tidak ada korelasi linier, kedua variabel kemungkinan besar tidak saling mempengaruhi.
- -1: Total korelasi linier negatif.

Pearson Correlation adalah metode default dari fungsi "corr". Seperti sebelumnya kita dapat menghitung Korelasi Pearson dari variabel 'int64' atau 'float64'. Terkadang kita ingin mengetahui signifikansi dari estimasi korelasi, kita dapat menggunakan p-value.

P-Value:

Berapa nilai P ini? Nilai P adalah nilai probabilitas bahwa korelasi antara kedua variabel ini signifikan secara statistik. Biasanya, kita memilih tingkat signifikansi 0,05, yang berarti bahwa kami yakin bahwa 95% korelasi antar variabel signifikan.

Dengan konvensi, ketika

- nilai p adalah $\leq 0,001$: kami katakan ada bukti kuat bahwa korelasinya signifikan.
- nilai p adalah $\leq 0,05$: terdapat bukti moderat bahwa korelasi tersebut signifikan.
- nilai p adalah $\leq 0,1$: ada bukti lemah bahwa korelasinya signifikan.
- nilai p adalah $> 0,1$: tidak ada bukti bahwa korelasi tersebut signifikan.

Kita dapat menggunakan library scipy untuk menghitung korelasi dan p-value

```
from scipy import stats
```

Mari kita hitung Koefisien Korelasi Pearson dan nilai-P dari 'wheel-base' dan 'price'.

```
pearson_coef, p_value = stats.pearsonr(df['wheel-base'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.584641822265508 with a P-value of P = 8.076488270733218e-20
```

Karena nilai p adalah $\leq 0,001$, korelasi antara wheel-base dan harga signifikan secara statistik, meskipun hubungan liniernya tidak terlalu kuat (0,588)

Mari kita hitung Koefisien Korelasi Pearson dan nilai-P dari 'horsepower' dan 'harga'.

```
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a
P-value of P =", p_value)
```

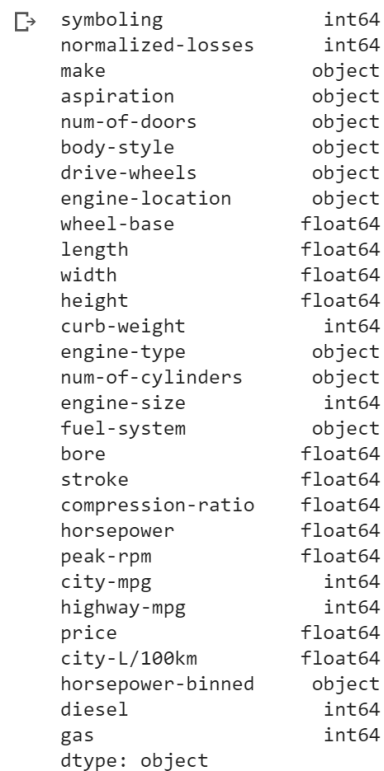
```
The Pearson Correlation Coefficient is 0.8095745670036559 with a P-value of P = 6.369057428260101e-48
```


Pertemuan 3

Karena nilai p adalah $< 0,001$, korelasi antara horsepower dan harga signifikan secara statistik, dengan korelasi linear positif yang cukup kuat ($\sim 0,805$)

Saat memvisualisasikan variabel individual, penting untuk terlebih dahulu memahami jenis variabel apa yang Anda hadapi (Gambar 0.28). Hal ini akan membantu kita menemukan metode visualisasi yang tepat untuk variabel tersebut.

```
# list the data types for each column  
print(df.dtypes)
```



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	float64
city-L/100km	float64
horsepower-binned	object
diesel	int64
gas	int64
dtype:	object

Gambar 0.28. Type data

misalnya, kita dapat menghitung korelasi antara variabel bertipe "int64" atau "float64" menggunakan method "corr" (Gambar 0.29):

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
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	0.279740	-0.035527	0.036233	-0.082391
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	0.239543	-0.225016	-0.181877	0.133999
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	-0.360305	-0.470606	-0.543304	0.584642
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	-0.285970	-0.665192	-0.698142	0.690628
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	-0.245800	-0.633531	-0.680635	0.751265
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	-0.309974	-0.049800	-0.104812	0.135486
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	-0.279361	-0.749543	-0.794889	0.834415
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	-0.256733	-0.650546	-0.679571	0.872335
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	-0.267392	-0.582027	-0.591309	0.543155
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	-0.065713	-0.034696	-0.035201	0.082310
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	-0.435780	0.331425	0.268465	0.071107
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	0.107885	-0.822214	-0.804575	0.809575
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	1.000000	-0.115413	-0.058598	-0.101616
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	-0.115413	1.000000	0.972044	-0.686571
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	-0.058598	0.972044	1.000000	-0.704692
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	-0.101616	-0.686571	-0.704692	1.000000
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	0.115830	-0.949713	-0.930028	0.789898
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	-0.475812	0.265676	0.198690	0.110326
gas	0.196735	0.101546	-0.307237	-0.211187	-0.244356	-0.281578	-0.221046	-0.070779	-0.054458	-0.241303	-0.985231	0.169053	0.475812	-0.265676	-0.198690	-0.110326

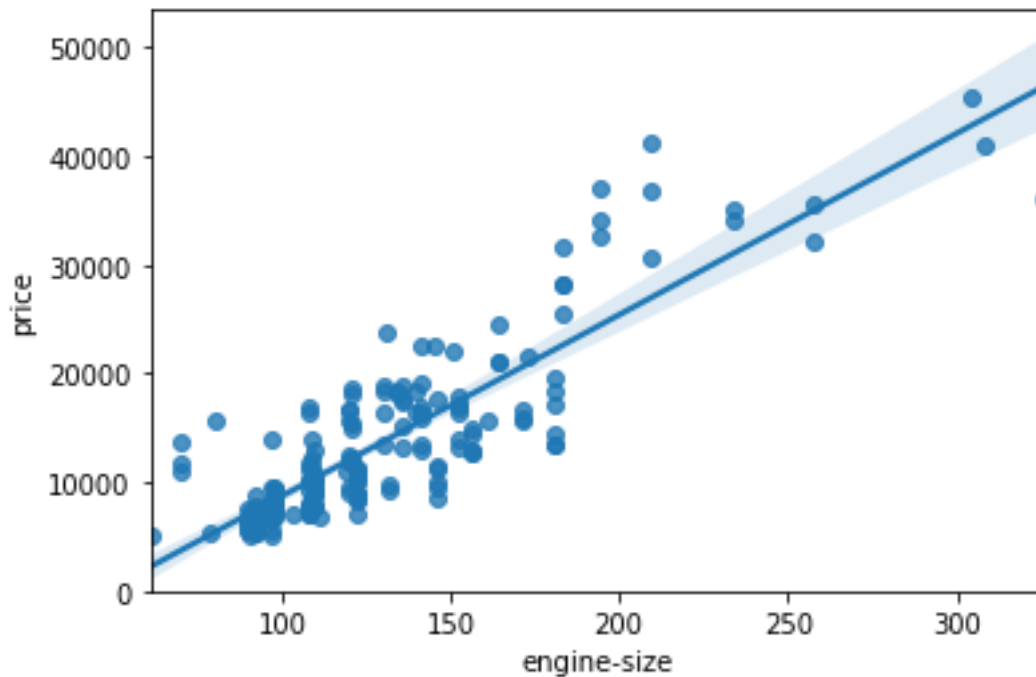
Gambar 0.29. Nilai korelasi antara varibel didalam dataset mobil

Variabel numerik kontinu adalah variabel yang mungkin berisi nilai nunerik dalam rentang tertentu. Variabel numerik kontinu dapat memiliki tipe "int64" atau "float64". Cara yang bagus untuk memvisualisasikan variabel-variabel ini adalah dengan menggunakan scatterplots dengan garis-garis yang pas.

Untuk mulai memahami keterhubungan (linier) antara variabel individu dan harga. Kita dapat melakukan ini dengan menggunakan "regplot". Fungsi ini yang memplot scatterplot ditambah garis regresi yang sesuai untuk data (Gambar 0.30).

Hubungan korelasi positif kuat antara variabel

```
# Engine size as potential predictor variable of price
sns.regplot(x="engine-size", y="price", data=df)
plt.ylim(0,)
```



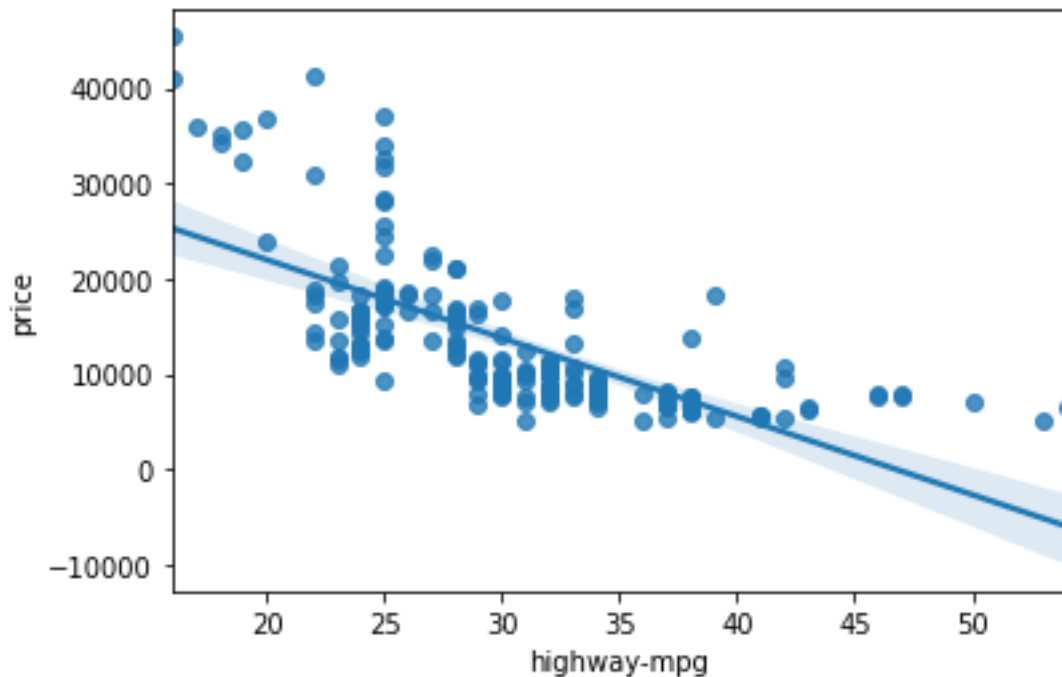
Gambar 0.30. Perbandingan korelasi antara engine-size dan price

Saat kapasitas mesin naik, harga mobil tersebut juga tinggi: ini menunjukkan hubungan linier antara kedua variabel tersebut. Ukuran mesin berpotensi menjadi prediktor harga. Kita dapat memeriksa korelasi antara engine-size dan harga sekitar 0,87

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

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

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



Gambar 0.31. Perbandingan antara variable highway-mpg dan harga

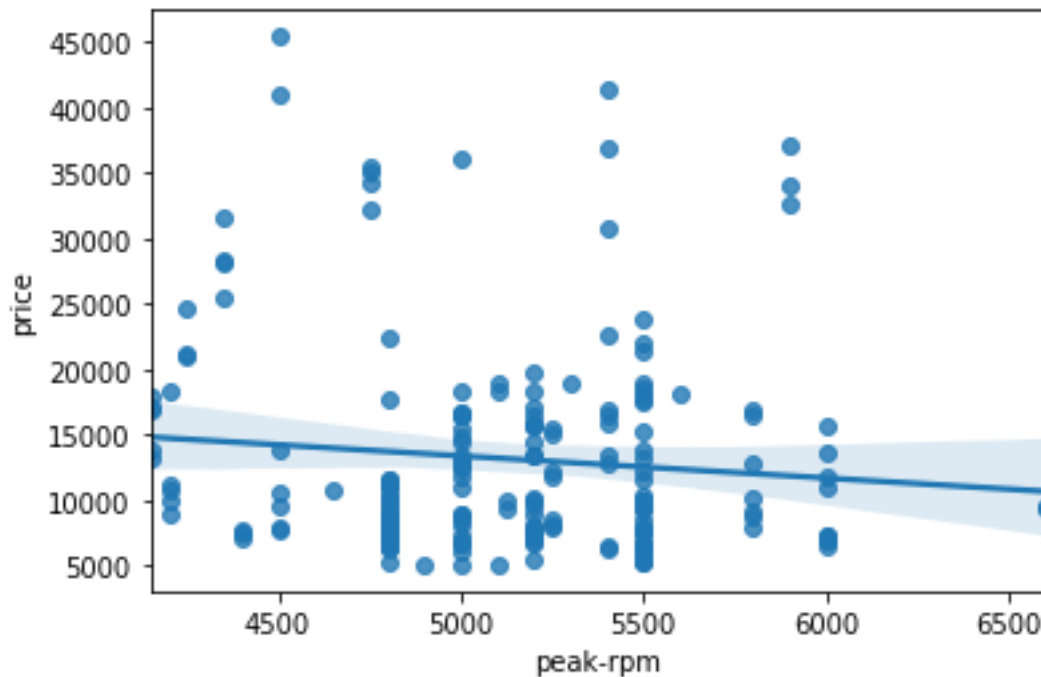
Saat highway-mpg naik, harganya mobil tersebut rendah: ini menunjukkan hubungan terbalik/negatif antara kedua variabel ini. Highway mpg berpotensi menjadi prediktor harga. Hal ini bisa dilihat sebagai korelasi kuat negative pada Gambar 0.31. Kita dapat memeriksa korelasi antara 'highway-mpg' dan 'price' adalah -0,704

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

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

Weak Linear Relationship

```
sns.regplot(x="peak-rpm", y="price", data=df)
```



Gambar 0.32. Perbandingan antara variable peak-rpm dan harga

Peak rpm sepertinya bukan merupakan prediktor harga yang baik karena garis regresinya mendekati horizontal (Gambar 0.32). Titik-titik data sangat tersebar dan jauh dari garis pas, menunjukkan banyak variabilitas. Oleh karena itu itu bukan variabel yang dapat diandalkan untuk memperdiksi harga. Kita dapat memeriksa korelasi antara 'puncak-rpm' dan 'harga' dan melihatnya kira-kira -0,101616

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

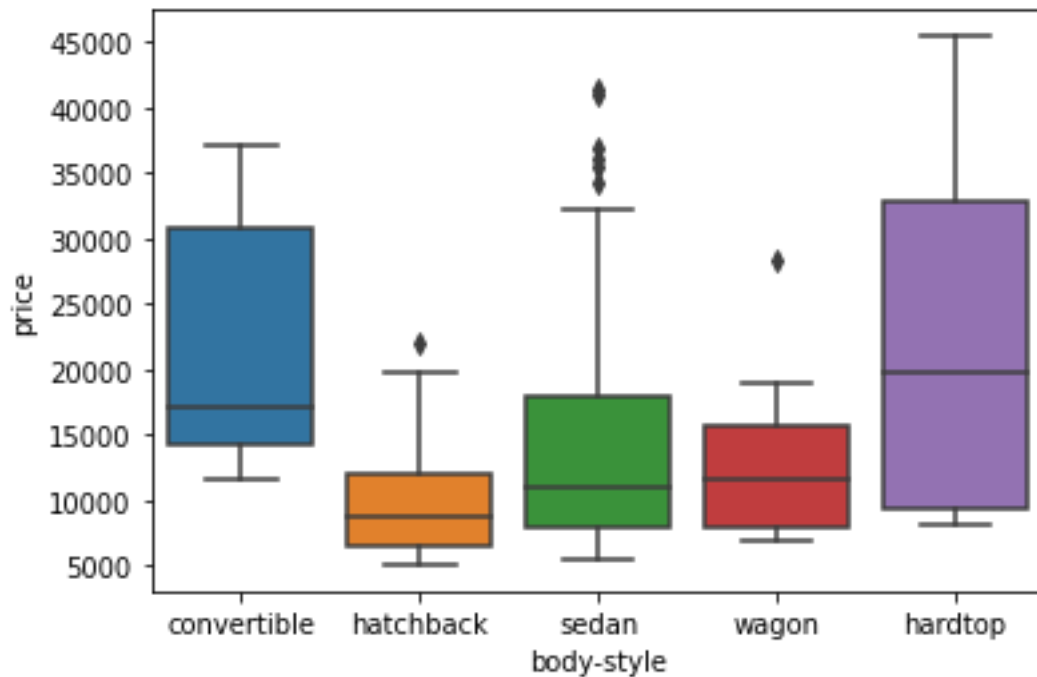
	peak-rpm	price
peak-rpm	1.000000	-0.101616
price	-0.101616	1.000000

Variabel Kategori Statistik

Variabel kategori statistic adalah variabel yang menggambarkan 'karakteristik' dari unit data, dan dipilih dari sekelompok kategori. Variabel kategori dapat memiliki tipe "objek" atau "int64". Cara yang baik untuk memvisualisasikan variabel kategori adalah dengan menggunakan boxplot.

Boxplot menggambarkan variable variable statistic seperti kuartil 1, median / kuartil 2, kuartil 3, nilai maksimum, nilai minimum, dan outlier (Gambar 0.33).

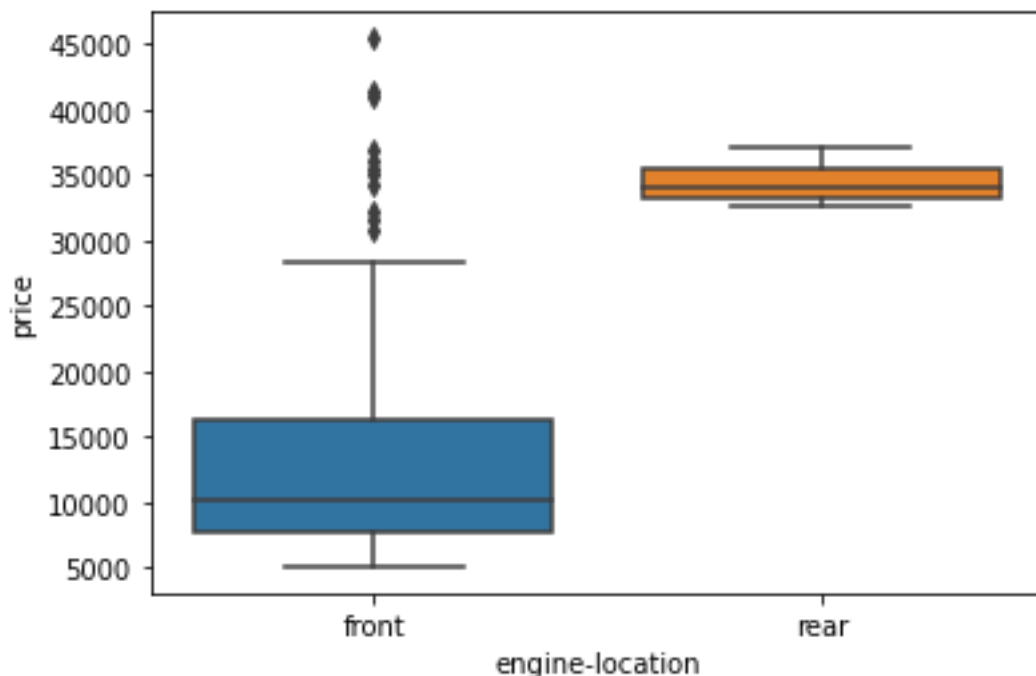
```
sns.boxplot(x="body-style", y="price", data=df)
```



Gambar 0.33. Contoh boxplot dari masing-masing jenis kendaraan

Kita melihat bahwa distribusi harga antara kategori kendaraan memiliki tumpang tindih yang signifikan, sehingga kategori tidak akan menjadi prediktor harga yang baik (Gambar 0.34). Mari kita periksa variable lokasi mesin dan harga:

```
sns.boxplot(x="engine-location", y="price", data=df)
```



Gambar 0.34. Perbandingan box-plot harga antara lokasi mesin di depan dan di belakang.

Visualisasi Deskriptif Statistik

Fungsi deskripsikan secara otomatis menghitung statistik dasar untuk semua variabel kontinu (Gambar 0.36). Analisis yang bisa kita dapatkan dari deskriptif statistik adalah

- Jumlah variabel
- Rata-rata
- Standard deviasi
- Nilai minimal
- IQR (Interquartile Range: 25%, 50% and 75%)
- Nilai Maximal

```
df.describe()
```

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	city-l/100km	diesel	gas
count	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	197.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000
mean	0.840796	122.000000	98.797015	0.837102	0.915126	53.766667	2555.666667	126.875622	3.330692	3.256904	10.164279	103.405534	5117.665368	25.179104	30.486567	13207.129353	9.944145	0.099502	0.900498
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	478.113805	6.423220	6.815150	7947.066342	2.534599	0.300083	0.300083
min	-2.000000	65.000000	86.600000	0.678039	0.837500	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000	4150.000000	13.000000	16.000000	5118.000000	4.795918	0.000000	0.000000
25%	0.000000	101.000000	94.500000	0.801538	0.890278	52.000000	2169.000000	98.000000	3.150000	3.110000	8.600000	70.000000	4800.000000	19.000000	25.000000	7775.000000	7.833333	0.000000	1.000000
50%	1.000000	122.000000	97.000000	0.832292	0.909722	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000	5125.369458	24.000000	30.000000	10295.000000	9.791667	0.000000	1.000000
75%	2.000000	137.000000	102.400000	0.881788	0.925000	55.500000	2926.000000	141.000000	3.580000	3.410000	9.400000	116.000000	5500.000000	30.000000	34.000000	16500.000000	12.368421	0.000000	1.000000
max	3.000000	256.000000	120.900000	1.000000	1.000000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	262.000000	6600.000000	49.000000	54.000000	45400.000000	18.076923	1.000000	1.000000

Gambar 0.35. Hasil descriptive statistics

Pengaturan default "describe" melewati variabel tipe objek. Kita bisa menggunakan code ini untuk menghitung jumlah type data objek (Gambar 0.36).

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

	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

Gambar 0.36. Hasil descriptive statistic untuk type data objek

Nilai-hitungan adalah cara untuk memahami berapa banyak unit dari setiap karakteristik/variabel yang kita miliki. Kita bisa menerapkan metode "value_counts" pada kolom 'drive-wheels'. Jangan lupa metode "value_counts" hanya berfungsi pada seri Pandas, bukan Pandas Dataframes.

```
df['drive-wheels'].value_counts()
fwd      118
rwd       75
4wd        8
Name: drive-wheels, dtype: int64
```

Kita dapat mengonversi seri ke Dataframe sebagai berikut:

Pertemuan 3

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

drive-wheels	
fwd	118
rwd	75
4wd	8

Mari ulangi langkah di atas tetapi simpan hasilnya ke dataframe "drive_wheels_counts" dan ganti nama kolom 'drive-wheels' menjadi 'value_counts'.

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

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

```
drive_wheels_counts
```

value_counts	
fwd	118
rwd	75
4wd	8

Sekarang mari kita ganti nama indeks menjadi 'drive-wheels':

```
drive_wheels_counts.index.name = 'drive-wheels'
```

```
drive_wheels_counts
```

value_counts	
drive-wheels	
fwd	118
rwd	75
4wd	8

Kita dapat mengulangi proses di atas untuk variabel 'engine-location'.

```
# engine-location as variable
```

```
engine_loc_counts = df['engine-location'].value_counts().to_frame()
```

```
engine_loc_counts.rename(columns={'engine-  
location': 'value_counts'}, inplace=True)
```

```
engine_loc_counts.index.name = 'engine-location'
```

```
engine_loc_counts.head(10)
```

value_counts	
engine-location	
front	198
rear	3

Memeriksa jumlah lokasi mesin mobil tidak akan menjadi variabel prediktor yang baik untuk harga. Karena, kita hanya punya tiga mobil dengan mesin belakang dan 198 dengan mesin di depan, hasilnya sangat tidak seimbang. Oleh karena itu, lokasi mesin bukan sebagai prediktor yang baik untuk harga.

Grouping

Method "groupby" digunakan untuk mengelompokkan data menurut kategori yang berbeda. Data dikelompokkan berdasarkan satu atau beberapa variabel dan analisis dilakukan pada kelompok individu.

Sebagai contoh, mari kita kelompokkan berdasarkan variabel "roda penggerak". Kita melihat bahwa ada 3 kategori roda penggerak yang berbeda.

```
df['drive-wheels'].unique()
```

```
array(['rwd', 'fwd', '4wd'], dtype=object)
```

Jika kita ingin mengetahui, secara rata-rata, jenis roda penggerak mana yang paling mahal, kita dapat mengelompokkan "roda penggerak" dan kemudian membuat rata-ratanya. Kita dapat memilih kolom 'drive-wheels', 'body-style' dan 'price', lalu menetapkan ke variabel "df_group_one".

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

Kami kemudian dapat menghitung harga rata-rata untuk setiap kategori data yang berbeda.

```
# grouping results
df_group_one = df_group_one.groupby(['drive-
wheels'], as_index=False).mean()
df_group_one
```

	drive-wheels	price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19757.613333

Dari data kita, sepertinya kendaraan roda belakang rata-rata paling mahal, sedangkan penggerak 4 roda dan roda depan harganya kurang lebih sama. Anda juga dapat mengelompokkan dengan beberapa variabel. Misalnya, mari kita kelompokkan berdasarkan 'roda penggerak' dan 'body-style'. Ini mengelompokkan dataframe dengan kombinasi unik 'drive-wheels' dan 'body-style'. Kita dapat menyimpan hasilnya dalam variabel 'grouped_test1'.

```
# grouping results
df_gptest = df[['drive-wheels', 'body-style', 'price']]
grouped_test1 = df_gptest.groupby(['drive-wheels', 'body-
style'], as_index=False).mean()
grouped_test1
```

	drive-wheels	body-style	price
0	4wd	hatchback	7603.000000
1	4wd	sedan	12647.333333
2	4wd	wagon	9095.750000
3	fwd	convertible	11595.000000
4	fwd	hardtop	8249.000000
5	fwd	hatchback	8396.387755
6	fwd	sedan	9811.800000
7	fwd	wagon	9997.333333
8	rwd	convertible	23949.600000
9	rwd	hardtop	24202.714286
10	rwd	hatchback	14337.777778
11	rwd	sedan	21711.833333
12	rwd	wagon	16994.222222

Data yang dikelompokkan ini jauh lebih mudah untuk divisualisasikan ketika dibuat menjadi tabel pivot. Tabel pivot yang mirip seperti pada spreadsheet Excel, dengan satu variabel di sepanjang kolom dan variabel lainnya di sepanjang baris. Kita dapat mengonversi kerangka data menjadi tabel pivot menggunakan metode "pivot" untuk membuat tabel pivot dari grup. Dalam hal ini, kita akan membiarkan variabel drive-wheel sebagai baris tabel, dan pivot body-style menjadi kolom tabel:

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

	price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	NaN	NaN	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

Seringkali, kita tidak memiliki data untuk beberapa sel pivot. Kita dapat mengisi sel yang hilang ini dengan nilai 0, tetapi nilai lain apa pun berpotensi digunakan juga. Harus disebutkan bahwa data yang hilang adalah subjek yang cukup kompleks.

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

	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

Gunakan fungsi "groupby" untuk mencari "harga" rata-rata setiap mobil berdasarkan "body-style" ?

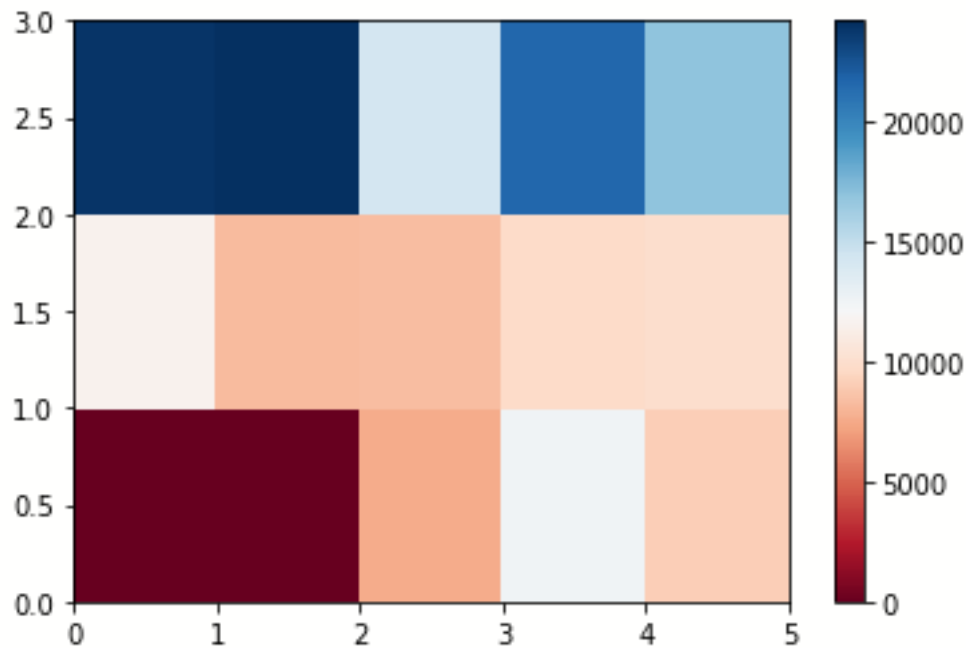
```
# Write your code below and press Shift+Enter to execute
df_gptest2 = df[['body-style', 'price']]
grouped_test_bodystyle = df_gptest2.groupby(['body-
style'], as_index= False).mean()
grouped_test_bodystyle
```

	body-style	price
0	convertible	21890.500000
1	hardtop	22208.500000
2	hatchback	9957.441176
3	sedan	14459.755319
4	wagon	12371.960000

Heatmap Grouping

Hasil dari pivot dapat kita visualisasikan dalam bentuk heatmap dapat kita lihat pada Gambar 0.37

```
import matplotlib.pyplot as plt
#use the grouped results
plt.pcolor(grouped_pivot, cmap='RdBu')
plt.colorbar()
plt.show()
```



Gambar 0.37. Heatmap grouping dari data kendaraan (group by roda penggerak)

Heatmap memplot variabel target (harga) dengan variabel 'roda penggerak' dan 'body-style' disumbu vertikal dan horizontal. Hal ini memungkinkan kita untuk memvisualisasikan bagaimana harga terkait dengan 'drive-wheel' dan 'body-style'.

Label default belum menyampaikan informasi yang cukup kepada kita. Mari kita ubah label pada heatmap tersebut agar bisa memiliki informasi legend (Gambar 0.38):

```
fig, ax = plt.subplots()
im = ax.pcolor(grouped_pivot, cmap='RdBu')

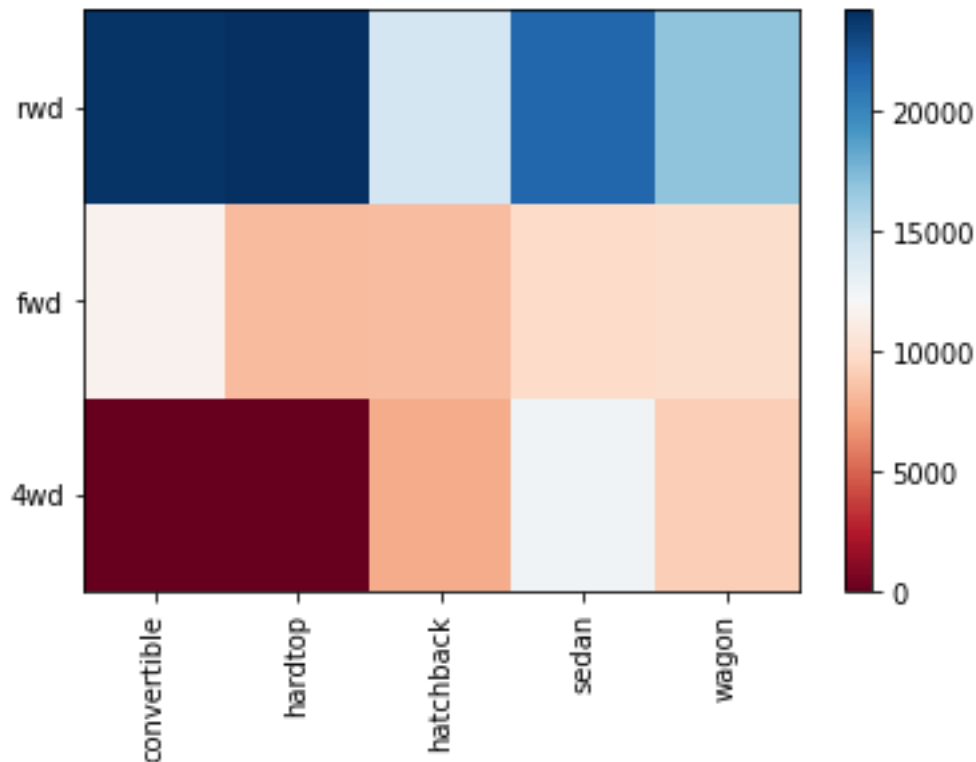
#label names
row_labels = grouped_pivot.columns.levels[1]
col_labels = grouped_pivot.index

#move ticks and labels to the center
ax.set_xticks(np.arange(grouped_pivot.shape[1]) + 0.5, minor=False)
ax.set_yticks(np.arange(grouped_pivot.shape[0]) + 0.5, minor=False)

#insert labels
ax.set_xticklabels(row_labels, minor=False)
ax.set_yticklabels(col_labels, minor=False)

#rotate label if too long
plt.xticks(rotation=90)

fig.colorbar(im)
plt.show()
```



Gambar 0.38. Heatmap grouping dari data kendaraan + legend (group by roda penggerak)

Visualisasi sangat penting dalam data science, dan paket visualisasi Python memberikan kebebasan untuk dapat dikonfigurasi. Pertanyaan utama yang ingin dijawab pada dataset ini, adalah "Apakah karakteristik utama yang paling berpengaruh terhadap harga mobil?".

ANOVA: Analysis of Variance

Analysis of Variance (ANOVA) adalah metode statistik yang digunakan untuk menguji apakah ada perbedaan yang signifikan antara rata-rata dua kelompok atau lebih. ANOVA mengembalikan dua parameter

F-Score: ANOVA mengasumsikan rata-rata semua kelompok adalah sama, anova akan menghitung seberapa jauh rata-rata yang sebenarnya menyimpang dari asumsi, dan melaporkannya sebagai F-Score. Skor yang lebih besar berarti ada perbedaan yang lebih besar antara rata-rata.

P-Value: Nilai-P menunjukkan seberapa signifikan secara statistik nilai skor yang dihitung.

Jika variabel harga pada dataset mobil sangat berkorelasi dengan variabel lainnya, ANOVA akan mengembalikan skor F-Score yang cukup besar dan nilai-p yang kecil.

ANOVA menganalisis perbedaan antara kelompok yang berbeda dari variabel yang sama, fungsi groupby akan berguna dalam kasus ANOVA.

Mari kita lihat apakah jenis 'roda penggerak' mempengaruhi 'harga',

Praktikum ST152-Big-Data-Predictive-Analysis

Pertemuan 3

```
grouped_test2=df_gptest[['drive-wheels', 'price']].groupby(['drive-wheels'])
grouped_test2.head(2)
```

	drive-wheels	price
0	rwd	13495.0
1	rwd	16500.0
3	fwd	13950.0
4	4wd	17450.0
5	fwd	15250.0
136	4wd	7603.0

```
df_gptest
```

	drive-wheels	body-style	price
0	rwd	convertible	13495.0
1	rwd	convertible	16500.0
2	rwd	hatchback	16500.0
3	fwd	sedan	13950.0
4	4wd	sedan	17450.0
...
196	rwd	sedan	16845.0
197	rwd	sedan	19045.0
198	rwd	sedan	21485.0
199	rwd	sedan	22470.0
200	rwd	sedan	22625.0

Kita dapat memperoleh nilai dari grup , method yang digunakan adalah "get_group".

```
grouped_test2.get_group('4wd')['price']
```

```
4      17450.0
136     7603.0
140     9233.0
141    11259.0
144     8013.0
145    11694.0
150     7898.0
151     8778.0
```

```
Name: price, dtype: float64
```

kita dapat menggunakan fungsi 'f_oneway' di modul 'stats' untuk mendapatkan F-Score dan P-Value

```
# ANOVA
```

Pertemuan 3

```
f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'],
grouped_test2.get_group('rwd')['price'], grouped_test2.get_group('4wd')
['price'])
```

```
print( "ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 67.95406500780399 , P = 3.3945443577151245e-23
```

Hasil ANOVA ini termasuk hasil yang bagus, dengan F-Score yang besar menunjukkan korelasi yang kuat dan nilai P hampir 0 menyiratkan signifikansi statistik yang hampir pasti.

Tetapi apakah ini berarti ketiga kelompok yang diuji semuanya berkorelasi tinggi?

Separately: fwd and rwd

```
f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'],
grouped_test2.get_group('rwd')['price'])
```

```
print( "ANOVA results: F=", f_val, ", P =", p_val )
```

```
ANOVA results: F= 130.5533160959111 , P = 2.2355306355677845e-23
```

4wd and fwd

```
f_val, p_val = stats.f_oneway(grouped_test2.get_group('4wd')['price'],
grouped_test2.get_group('fwd')['price'])
```

```
print("ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 0.665465750252303 , P = 0.41620116697845666
```
