**Cara Binning Data di Python dengan Pandas**

Saat kita sedang menangani data numerik yang bersifat kontinu, masalah yang sering dihadapi adalah dalam mengamati sebaran nilainya atau distribusi datanya. Bagaimana cara mengatasi permasalahan tersebut. Nah, untuk memudahkan  proses analisis data, kita dapat melakukan binning data.

**Apa itu binning data?**

Binning adalah sebuah proses untuk mengelompokkan data ke dalam bagian-bagian yang lebih kecil yang disebut bin berdasarkan kriteria tertentu. Binning data merupakan salah satu teknik praproses data yang digunakan untuk meminimalisasi kesalahan dalam pengamatan serta terkadang dapat meningkatkan akurasi dari model prediktif.

Binning biasanya digunakan untuk mengelompokkan data numerik menjadi beberapa bin agar sebaran data lebih mudah dipahami. Misalnya kita dapat mengelompokkan fitur “usia” menjadi [0-5], [6-10], [11-15], [16-20], [20-25], dan seterusnya.

Contoh lain misalnya sebuah data yang memiliki fitur “harga” dengan kisaran 1000 hingga 99000. Kita bisa membagi harga tersebut menjadi tiga bin, “murah”, “standar”, dan “mahal” dengan batas tertentu.

Di Python, ada beberapa cara yang dapat dilakukan untuk membagi data menjadi beberapa bin.

**Cara binning data di Python**

Disini saya akan membahas dua cara yang dapat kita lakukan untuk melakukan binning data di Python dengan Pandas. Ada dua fungsi di Pandas yang dapat digunakan untuk melakukan binning data, yaitu cut() dan qcut().

**cut()**

Yang pertama akan kita bahas adalah fungsi cut(). Misalnya kita memiliki data harga dengan range 1500 sampai 91000 dan terdiri dari 12 item seperti pada tabel berikut.

| **Item** | **Harga** |
| --- | --- |
| Item\_1 | 1500 |
| Item\_2 | 3300 |
| Item\_3 | 11000 |
| Item\_4 | 87500 |
| Item\_5 | 45000 |
| Item\_6 | 28600 |
| Item\_7 | 39900 |
| Item\_8 | 91000 |
| Item\_9 | 64700 |
| Item\_10 | 6000 |
| Item\_11 | 19000 |
| Item\_12 | 2700 |

Kita akan mengategorikan harga 12 item tersebut menjadi 3 bin yaitu murah, standar, dan mahal.

Langkah pertama yang harus dilakukan adalah menentukan batas minimum dan maksimum untuk tiap kategori. Ada 3 cara yang dapat kita lakukan yaitu dengan menggunakan linspace() yang merupakan fungsi dari *library* Numpy, menggunakan interval\_range(), dan yang terakhir dengan menentukan sendiri angka batasan yang diinginkan.

**Menggunakan linspace()**

linspace() adalah sebuah fungsi dari Numpy yang digunakan untuk membuat array yang berisi sejumlah angka dengan jarak yang sama. Angka-angka ini yang nantinya akan digunakan sebagai batas untuk tiap kategori. Secara umum di Python linspace() dapat ditulis sebagai berikut.

numpy.linspace(nilai minimum, nilai maksimum, jumlah elemen)

Dalam kasus “harga”, nilai minimum berarti harga terendah di dalam data, harga maksimum berarti harga tertinggi. Sementara itu, jumlah elemen disini berarti mau dibagi menjadi berapa bagian data tersebut. Karena kita akan membagi menjadi 3 kategori, maka kita membutuhkan 4 elemen atau angka sebagai batasnya.

Di Python, kita dapat membuat kodenya seperti ini.

bins = np.linspace(min(data['Harga']), max(data['Harga']), 4)

Untuk melihat batasan angkanya, kita dapat print bins.

print(bins)

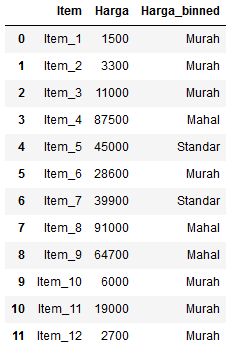
Selanjutnya kita definisikan kategori harganya.

kategori = ['Murah', 'Standar', 'Mahal']

Setelah menentukan batas dan kategori, kita dapat melakukan binning pada data tersebut menggunakan fungsi Pandas cut() dengan kode seperti di bawah ini.

data['Harga\_binned'] = pd.cut(data['Harga'], bins, labels=kategori, include\_lowest=True)

Hasil yang diperoleh dari kode di atas adalah sebagai berikut.



Terlihat ada 7 item yang termasuk kategori Murah, 2 item di kategori Standar, dan 3 item di kategori Mahal.

**Menggunakan interval\_range()**

Kita juga dapat menggunakan interval\_range() yang merupakan sebuah fungsi dari Pandas untuk menentukan batasan bin. Jika kita terapkan pada data harga, kodenya dapat ditulis sebagai berikut.

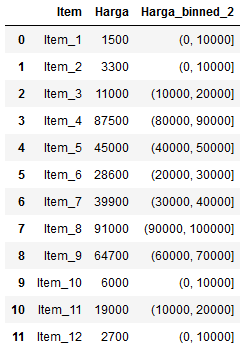
interval\_range = pd.interval\_range(start=0, freq=10000, end=100000)

Parameter start merupakan nilai awal, end merupakan nilai akhir, dan freq merupakan panjang dari tiap interval. Untuk lebih detailnya tentang interval\_range() dapat dilihat [disini](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.interval_range.html).

Selanjutnya kita gunakan fungsi cut() untuk binning data.

data['Harga\_binned\_2'] = pd.cut(data['Harga'], bins=interval\_range)

Hasilnya kira-kira seperti ini.



Harap dicatat bahwa jika menggunakan interval\_range(), kita tidak dapat memberikan label (Murah, Standar, Mahal).

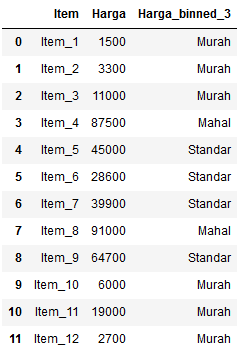
Untuk membaca interval seperti **(0, 10000]**, **(10000, 20000]**, dan sebagainya, tanda kurung () berarti tidak diikutsertakan, sedangkan tanda kurung siku [] berarti diikutsertakan. Jadi untuk contoh interval **(10000, 20000]** misalnya, 10000 tidak masuk dalam interval, tetapi 20000 masuk dalam interval tersebut.

**Mendefinisikan sendiri batasan bin**

Kita juga dapat mendefinisikan batas-batas bin yang akan dibuat sesuai keinginan. Di Python, dapat ditulis sebagai berikut.

batas\_bin = [0, 20000, 70000, 100000]  
kategori = ['Murah', 'Standar', 'Mahal']  
data['Harga\_binned\_3'] = pd.cut(data['Harga'], bins=batas\_bin, labels=kategori)

Hasilnya kira-kira seperti ini.



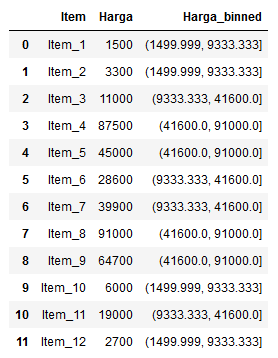
**qcut()**

Selain fungsi cut(), ada juga fungsi qcut() yang dapat digunakan untuk melakukan binning data. Menurut dokumentasi pandas, qcut digambarkan sebagai *Quantile-based discretization function*. Singkatnya fungsi qcut() ini akan membagi data ke dalam jumlah yang sama. Karena itu, jarak untuk masing-masing bin boleh jadi berbeda satu sama lain.

Di Python, jika kita akan melakukan binning data menjadi 3 bin menggunakan qcut() dapat ditulis sebagai berikut.

data['Harga\_binned'] = pd.qcut(data['Harga'], 3)

Hasilnya seperti di bawah ini.

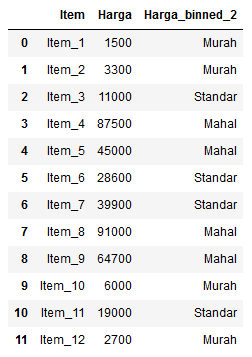


Seperti dapat kita lihat di atas, jarak atau panjang tiap interval berbeda-beda, namun memiliki jumlah yang sama, dimana masing-masing bin terdiri dari 4 item. Kita juga dapat menambahkan label kategori dengan terlebih dahulu mendefinisikan kategorinya, kemudian menambahkan parameter labels pada qcut().

kategori = ['Murah', 'Standar', 'Mahal']

data['Harga\_binned\_2'] = pd.qcut(data['Harga'], q=3, labels=kategori)

Hasilnya kira-kira seperti ini.



Kesimpulannya, jika kita mengharapkan jumlah item yang sama pada masing-masing bin, gunakanlah qcut(). Namun, jika lebih mengutamakan batas angka tertentu untuk tiap bin, gunakanlah cut().