Nah pada submodul ini kita akan membahas lebih lanjut mengenai feature engineering. *Andrew Ng*, seorang profesor kecerdasan buatan dari *Stanford* dan pencetus *Google Brain* mengatakan bahwa “Menciptakan fitur-fitur yang baik adalah pekerjaan yang sulit, memakan waktu, dan membutuhkan pengetahuan seorang pakar di bidang terkait. Machine learning terapan pada dasarnya adalah rekayasa fitur.”

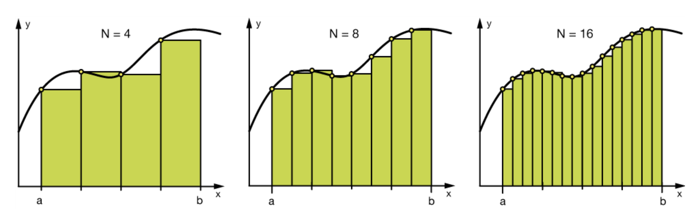
Berdasarkan pernyataan dari *Andrew* maka dapat disimpulkan, bahwa tahap rekayasa fitur merupakan salah satu tahapan yang memakan banyak waktu. Jadi, rekayasa fitur itu pastinya penting ya.

Nah, di submodul ini Anda akan dikenalkan dengan beberapa teknik lain dalam rekayasa fitur yang sudah dibahas pada modul sebelumnya.

**Binning**

Binning adalah pengelompokan nilai sesuai dengan batas besaran yang ditentukan. Pada binning, data dikelompokkan dalam tiap ‘**bin**’ sesuai dengan nilai yang cocok dengan bin tersebut. Bin sederhananya adalah sebuah kategori yang menampung nilai-nilai tertentu.

Ada beberapa jenis binning di mana salah satu contohnya adalah binning jarak. Pada binning jarak, nilai-nilai dari sebuah atribut akan dikategorikan ke dalam jumlah bin tertentu yang memiliki interval sama besar. Pada gambar dibawah dapat dilihat contoh kumpulan nilai yang dibagi menjadi **4 bin**, **8 bin**, dan **16 bin**.

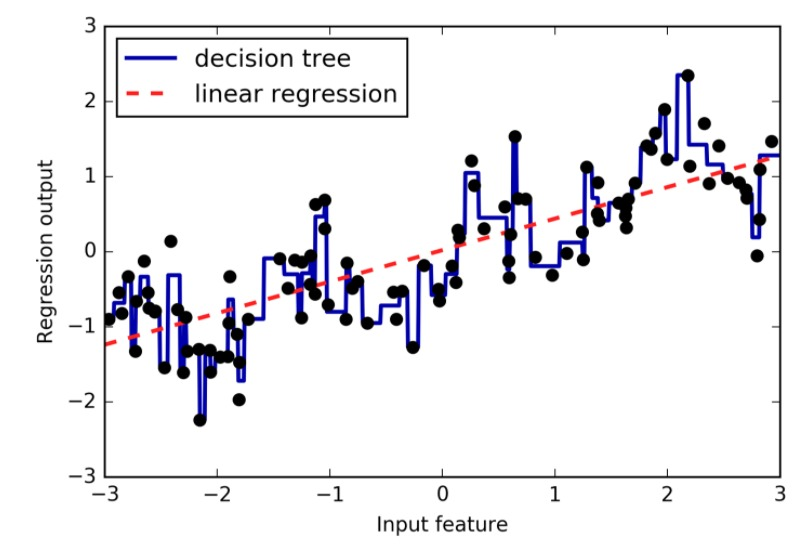


Binning bisa membantu untuk menghindari *overfitting*. Namun binning juga mengorbankan informasi yang terkandung dari sebuah atribut sehingga, penggunaanya perlu dilakukan dengan teliti. Di bawah adalah contoh untuk melakukan binning pada **dataframe Pandas**.

1. data[‘bin’] = pd.cut(data[‘value’], bins=[0, 30,  70, 100], labels=[“Low”, “Mid”, “High”])

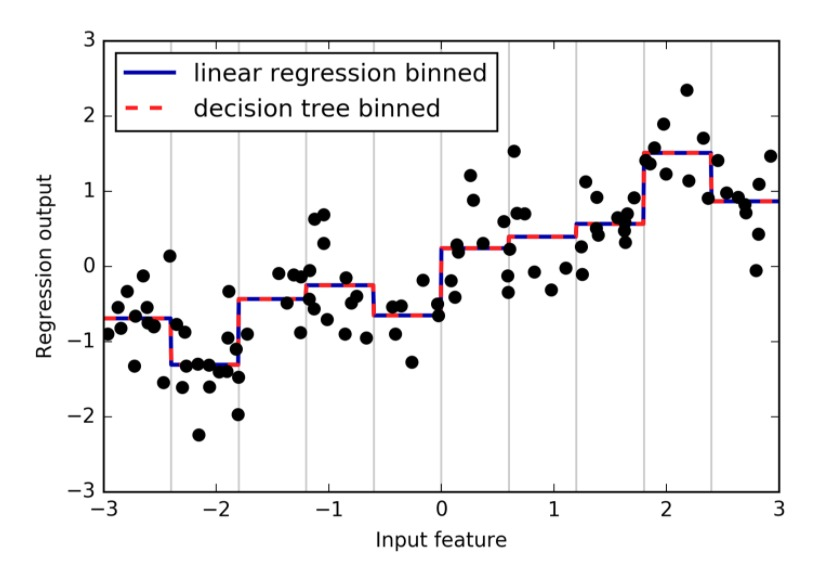
4. value    bin
5. 0      13    Low
6. 1      25    Low
7. 2      32    Mid
8. 3      94    High
9. 4      49    Mid

Mari kita ulas aplikasi binning pada model linear seperti yang dituliskan oleh S. Guido dan A. C. Muller dalam bukunya [1]. Seperti yang kita tahu, model linear hanya bisa menghasilkan relasi linear, yang mana hasil regresinya merupakan garis pada fitur tunggal. Di sisi lain, model decision tree dapat menghasilkan model yang lebih kompleks dari data, meskipun model ini sangat bergantung pada representasi data. Salah satu cara untuk membuat model linear lebih efektif dalam data kontinyu adalah dengan melakukan proses binning pada fitur.



Gambar di atas menunjukkan perbandingan regresi linear dan decision tree pada dataset gelombang. Dengan teknik binning, kita membagi jangkauan input untuk fitur (dalam hal ini dari -3 ke 3) ke dalam angka/bin yang tetap, misal 10. Titik-titik data kemudian direpresentasikan ke dalam bin yang telah ditentukan tadi. Ada beberapa cara untuk menentukan batas bin, misalnya dengan membuat tepi bin berjarak sama, atau menggunakan kuartil data. Pada library sklearn, kedua teknik ini diimplementasikan dengan KBinsDiscretizer.

Jika kita melakukan proses binning pada data gambar di atas, maka hasilnya akan seperti berikut.



Kalau Anda tertarik untuk mempelajari lebih lanjut tentang binning, kunjungi tautan-tautan berikut yah, [tautan 1](https://www.coursera.org/lecture/data-analysis-with-python/binning-in-python-T8z3M), [tautan 2](https://developers.google.com/machine-learning/data-prep/transform/bucketing), [tautan 3](https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/group-data-into-bins).

**Splitting**

Memisahkan sebuah atribut menjadi atribut-atribut baru juga merupakan salah satu cara yang berguna. Dengan splitting kita membuat atribut lebih mudah dipahami oleh model machine learning. Di lapangan sering kita temui data dengan kolom string melanggar prinsip [tidy data](https://www.jstatsoft.org/article/view/v059i10) (Hadley Wickham). Memisahkan sebagian data dalam kolom menjadi fitur baru memberikan keuntungan antara lain: 1) meningkatkan performa model dengan menemukan informasi berharga; dan 2) membuat kita lebih mudah untuk melakukan proses binning dan grouping.

Ada beberapa cara untuk melakukan fungsi split, tergantung pada karakteristik kolom. Mari kita ambil salah satu contoh kasus. Sebuah atribut dengan judul **“ram\_hardisk”** yang berisi informasi mengenai besar ram dan penyimpanan dari harddisk. Kita dapat memisahkan atribut tersebut menjadi **“ram”** dan**“storage**” untuk memudahkan model mendapatkan informasi lebih banyak dari atribut baru.

Contoh lain adalah kolom **‘full\_name’**. Misal kita hanya membutuhkan informasi nama hanya terdiri dari satu kata. Maka, kita dapat membagi atribut pada kolom nama dengan memisahkan data **‘first\_name’** dan **‘last\_name’**kemudian menggunakan salah satu atribut baru tersebut sesuai kebutuhan.

Perhatikan contoh code berikut ya.

1. import pandas as pd
3. Developer = pd.DataFrame({'Name': ['Isyana Saraswati', 'Nicholas Saputra', 'Raisa Andriana'],
4. 'Age':[30, 36, 32]})
5. print("Machine Learning Developer di Indonesia :\n", Developer)
7. Developer[['First','Last']] = Developer.Name.str.split(expand=True)
8. print("\n Split kolom 'Name', lalu tambahkan kedua kolom : \n", Developer)

Jika dieksekusi, code di atas akan memberikan hasil sebagai berikut.



**Interaction feature**

Dalam model machine learning, kita juga bisa menciptakan fitur atau atribut baru dari atribut-atribut yang ada. Mengkombinasikan dua atribut kadang bisa menjelaskan varian data dengan lebih baik dibanding dua atribut yang dioperasikan secara terpisah. Membuat atribut baru melalui interaksi antar atribut disebut sebagai interaction feature. Sederhananya, interaction feature adalah perkalian produk antara dua buah fitur. Analoginya adalah logika AND.

Mari kita jabarkan penjelasan di atas dalam contoh berikut.

* Misal kita telah memiliki sebuah atribut bernama **‘schools’**yaitu sejumlah sekolah yang berada dalam radius 5 km dari pusat kota.
* Kita juga memiliki atribut lain yaitu **‘accredited\_A’** yang merupakan sejumlah sekolah yang telah terakreditasi A.
* Dalam menentukan pilihan sekolah, kita mungkin menginginkan sekolah yang dekat tapi juga telah memiliki akreditasi A.
* Untuk mengakomodasi kebutuhan tersebut, kita dapat membuat atribut baru misalnya: **‘selected\_schools’ = ‘schools’ x ‘accredited\_A’**

Mudah bukan? Betul, interaction feature sangat mudah untuk dirumuskan, tetapi biaya komputasinya cukup tinggi. Untuk sebuah model linear dengan interaction feature berpasangan, kebutuhan waktu pelatihannya akan berubah dari O(n) menjadi O(n2), di mana n adalah jumlah fitur tunggal.

Ada beberapa cara untuk mengatasi permasalahan ini. Pertama adalah dengan melakukan feature selection, teknik lain dalam feature engineering. Cara kedua adalah dengan menyusun*(handcrafted)*sejumlah kecil fitur kompleks secara hati-hati.