

## Pendahuluan Studi Kasus Pertama: Predictive Analytics

Peningkatan penggunaan internet di seluruh dunia menghasilkan data dalam jumlah yang sangat besar. Berbagai perangkat cerdas dan internet membuat data menjadi berlimpah. Perusahaan dan organisasi mengumpulkan data berjumlah besar untuk berbagai kepentingan, salah satunya proses pengambilan keputusan.

Supaya memberikan manfaat dan dapat digunakan untuk membuat keputusan, data harus melalui proses analisis dan ekstraksi informasi (*insight*). Mengextrak informasi dari data adalah inti dari pekerjaan data analytics. Predictive analytics, pokok bahasan dalam modul ini merupakan sub-bidang data analytics.

Dalam artikel berjudul "*Competing on Analytics*" yang diterbitkan oleh [Harvard Business Review](#), Davenport, seorang ahli business analytics berpendapat bahwa senjata strategis di bidang bisnis saat ini adalah pengambilan keputusan analytics. Ia merupakan teknik pengambilan keputusan berdasarkan berbagai informasi yang diekstrak dari data.

Bagaimana data diubah menjadi informasi kemudian menghasilkan keputusan dilustrasikan dalam gambar berikut [19].



Saat mendengar istilah data analytics, Anda mungkin bertanya-tanya, apa bedanya istilah analytics dan analysis

Meskipun kedua istilah ini sering digunakan secara bergantian (*interchangeably*), keduanya tidaklah sama. Delen dalam "*Predictive Analytic: Data Mining, Machine Learning and Data Science for Practitioners*" [20] menyatakan, analisis mengacu pada proses pemisahan seluruh masalah menjadi bagian-bagian untuk diperiksa secara rinci dan –jika perlu– diperbaiki. Setelah proses pemeriksaan dan perbaikan selesai, seluruh sistem dapat dipadukan kembali.

Sementara itu, analytics mencakup berbagai metode, teknologi, dan alat untuk menciptakan informasi baru demi memecahkan permasalahan kompleks serta membuat keputusan yang lebih baik. Analytics merupakan pendekatan multidisiplin yang memanfaatkan data dan model matematika untuk memahami berbagai situasi yang kompleks. Proses analisis merupakan salah satu cakupan dari tahapan analytics.

Pertanyaan selanjutnya adalah, apa itu predictive analytics? Predictive analytics adalah bidang terapan yang menggunakan berbagai metode kuantitatif untuk membuat prediksi dengan memanfaatkan data. Ia merupakan seni membangun serta menggunakan model prediksi berdasarkan pola data historis. Diluktip dari Delen [20], beberapa contoh penggunaan predictive analytics, antara lain:

- **Prediksi Harga**

Organisasi bisnis seperti jaringan hotel, maskapai penerbangan, dan bisnis penjualan daring perlu terus-menerus menyesuaikan harga mereka untuk memaksimalkan keuntungan. Perusahaan harus bisa meningkatkan atau mempertahankan pendapatan dan profit meski menghadapi situasi yang sulit. Situasi sulit ini bisa disebabkan oleh berbagai faktor, antara lain perubahan musim, pergeseran permintaan pelanggan, dan terjadinya kejadian khusus seperti pandemi.

Model predictive analytics dapat dilatih untuk memprediksi harga optimal berdasarkan data atau catatan historis penjualan. Perusahaan kemudian dapat menggunakan prediksi ini sebagai masukan untuk menentukan berbagai keputusan bisnis, misalnya strategi penetapan harga.

- **Prediksi Dosis**

Dokter dan ilmuwan menentukan berapa banyak obat atau zat kimia yang diperlukan dalam pengobatan. Model predictive analytics dapat digunakan untuk membantu pengambilan keputusan dengan memprediksi dosis optimal berdasarkan data dosis masa lalu dan hasil terkait. Prediksi dosis dengan model predictive analytics tentu harus dibuat dengan sangat hati-hati demi keselamatan pasien.

- **Penilaian Risiko**

Risiko merupakan salah satu hal yang sangat berpengaruh dalam setiap keputusan organisasi. Model predictive analytics dapat digunakan untuk memprediksi risiko yang terkait dengan keputusan, seperti mengeluarkan pinjaman atau menanggung polis asuransi. Model ini dilatih menggunakan data historis yang berkaitan dengan indikator utama risiko. Keluaran dari model prediksi risiko dapat digunakan oleh organisasi untuk membuat penilaian risiko yang lebih baik di masa mendatang.

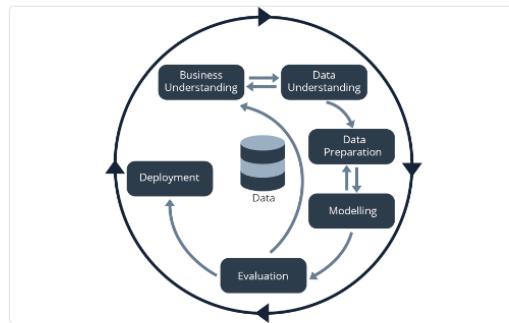
Bagaimana, Anda tertarik untuk membuat model predictive analytics? Di modul kali ini Anda akan mempelajari kasus predictive analytics pada bidang marketing. Hingga akhir modul ini diharapkan Anda dapat:

- Memahami problem statement pada kasus prediksi harga diamonds (berlian).
- Melakukan proses exploratory data analysis.
- Melakukan tahap data preparation.
- Membuat model machine learning dengan pendekatan K-Nearest Neighbor.
- Membuat model machine learning dengan pendekatan Random Forest.
- Membuat model machine learning dengan pendekatan Boosting Algorithm.
- Mengevaluasi Model.

Yuk, lanjut ke materi berikutnya!

## Predictive Analytics Lifecycle

Seperti proyek machine learning lainnya, proyek predictive analytics memiliki beberapa tahapan dalam proses pengembangannya. Keberhasilan proyek predictive analytics tidak hanya ditentukan oleh pemilihan algoritma machine learning saja, melainkan juga penerapan metodologi standar dalam mengelola seluruh tahapan atau siklus proyek. *Cross-Industry Standard Process for Data Mining* atau disingkat menjadi CRISP-DM merupakan salah satu metode standar proses analitik yang paling umum digunakan. Metode ini akan membantu Anda mengelola proyek dari tahapan mendefinisikan masalah hingga mendapatkan insight.



Dalam metode ini, proses analitik dibagi menjadi enam fase utama, antara lain:

### 1. Business understanding

Proyek predictive analytics biasanya berfokus pada hal-hal seperti mendapatkan pelanggan baru, meningkatkan performa penjualan, dan menambahkan efisiensi pada suatu proses tertentu. Fase awal dari proyek analitik bertujuan untuk memahami masalah bisnis kemudian merancang solusi analitik berdasarkan data untuk menyelesaikan permasalahan tersebut. Tahapan ini mencakup proses klarifikasi masalah, menentukan tujuan atau objective, dan mengidentifikasi sumber daya yang dimiliki.

### 2. Data understanding

Memiliki pemahaman mengenai data yang tersedia berikut sumbernya merupakan fase yang penting setelah tujuan proyek diputuskan. Fase ini memungkinkan Anda untuk memahami informasi dalam data dan menentukan kualitasnya. Pemahaman data sangat penting untuk menghindari masalah yang tidak terduga selama fase berikutnya, yaitu fase persiapan data (data preparation).

### 3. Data preparation

Data preparation merupakan salah satu tahapan penting yang paling memakan waktu. Proses yang Anda lakukan dalam tahapan ini bisa berbeda tergantung kebutuhan dan tujuannya. Namun, secara umum, hal-hal yang akan Anda lakukan dalam fase ini adalah sebagai berikut:

- Menggabungkan data.
- Menyeleksi data yang akan digunakan.
- Melakukan proses transformasi data
- Membagi data menjadi data training dan test.

### 4. Modeling

Pemodelan biasanya dilakukan dalam beberapa iterasi. Pada fase ini, Anda menggunakan algoritma machine learning yang berbeda untuk membuat model prediksi. Model-model ini kemudian dibandingkan dan model terbaik yang akan dipilih untuk diterapkan.

### 5. Evaluation

Sebelum model diterapkan, model perlu dievaluasi agar terbukti cocok untuk tujuan yang telah ditentukan. Fase ini bertujuan untuk memastikan bahwa model akan mampu membuat prediksi yang akurat dan tidak mengalami overfitting atau underfitting.

### 6. Deployment

Pada fase ini, model machine learning yang telah Anda buat akan diintegrasikan ke dalam sistem organisasi. Secara umum, proses yang Anda lakukan dalam fase ini adalah:

- Mengintegrasikan model ke dalam sistem (proses deployment).
- Memonitor hasil deployment.
- Membuat laporan akhir dan melakukan evaluasi proyek.

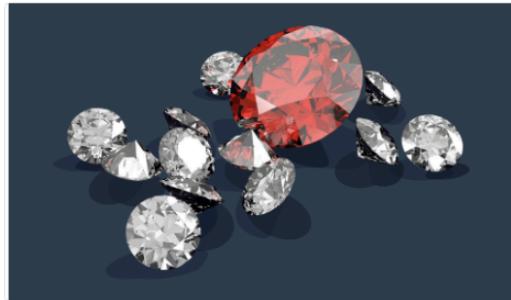
Setiap tahapan dalam metode ini penting untuk keberhasilan proyek. Urutan tahapan dalam metode ini menunjukkan hubungan yang paling sering antar fase. Ia membantu Anda dalam melihat skema garis besarnya serta memahami bagaimana setiap tahapan memberikan kontribusi pada tahapan berikutnya. Namun, tahapan-tahapan ini tidak harus dilukti secara berurutan. Seperti halnya sebuah siklus, Anda dapat mengulangi setiap tahapan jika diperlukan. Biasanya ini terjadi jika Anda menemukan informasi yang bertentangan dengan apa yang telah Anda temukan sebelumnya.

Sebagai contoh, saat membuat proyek machine learning untuk kasus predictive analytics, Anda merasa telah selesai dengan tahap persiapan data. Namun, saat melakukan proses pemodelan Anda menemukan fakta bahwa data Anda tidak dapat mengakomodasi kondisi tertentu. Oleh karena itu, Anda perlu kembali ke tahapan sebelumnya dan melakukan proses persiapan data kembali.

Selanjutnya, Anda akan langsung praktik membuat proyek machine learning untuk kasus predictive analytics menggunakan diamond price dataset. Yuk, lanjut ke materi berikutnya!

## Business Understanding

Bayangkan Anda adalah pemilik perusahaan yang bergerak di bidang jual-beli diamonds (berlian). Model bisnis Anda adalah distributor dan retail, perusahaan membeli diamonds dari produsen kemudian menjualnya kepada konsumen. Perusahaan juga menerima penjualan kembali diamonds dari konsumen. Untuk efisiensi, Anda ingin menerapkan automasi pada sistem dalam memprediksi harga diamonds dengan teknik *predictive modelling*.



Contoh kasus, Anda mengetahui bahwa harga sebuah diamonds dengan karakteristik tertentu bernilai \$9000 di pasaran. Jika ingin mendapatkan profit, tentu perusahaan harus mendapatkan harga beli diamond yang lebih rendah dari \$9000. Sebut saja misal, harga belinya adalah \$5000. Sudah pasti ini akan menjadi keuntungan besar bagi perusahaan. Sebaliknya, jika perusahaan membeli diamonds tersebut dengan harga di atas \$9000, maka perusahaan akan rugi.

Tentu saja semua bisnis mengejar profit. Oleh karena itu, penting bagi perusahaan untuk mengetahui dan dapat memprediksi harga diamonds di pasar. Prediksi akan digunakan untuk menentukan berapa harga beli yang pantas untuk diamonds dengan karakteristik tertentu sehingga perusahaan bisa mendapatkan profit sebesar mungkin.

Tidak seperti emas yang harga jual dan belinya mengacu pada harga perdagangan emas dunia, harga diamonds dipengaruhi oleh beberapa fitur khusus. Fitur tersebut antara lain, karat, ukuran, bentuk potongan, warna, serta tingkat kejernihan diamonds. Tidak adanya acuan harga diamonds seperti acuan harga emas menyebabkan perusahaan memerlukan sistem untuk memprediksi harganya.

## Problem Statements dan Goals

Berdasarkan kondisi yang telah diuraikan sebelumnya, perusahaan akan mengembangkan sebuah sistem prediksi harga diamonds untuk menjawab permasalahan berikut.

- Dari rangkaian fitur yang ada, fitur apa yang paling berpengaruh terhadap harga diamonds?
- Berapa harga pasar diamonds dengan karakteristik atau fitur tertentu?

Untuk menjawab pertanyaan tersebut, Anda akan membuat predictive modelling dengan tujuan atau goals sebagai berikut:

- Mengetahui fitur yang paling berkorelasi dengan harga diamonds.
- Membuat model machine learning yang dapat memprediksi harga diamonds seakurat mungkin berdasarkan fitur-fitur yang ada.

## Metodologi

Prediksi harga adalah tujuan yang ingin dicapai. Seperti yang kita tahu, harga merupakan variabel kontinu. Dalam predictive analytics, saat membuat prediksi variabel kontinu artinya Anda sedang menyelesaikan permasalahan regresi. Oleh karena itu, metodologi pada proyek ini adalah: membangun model regresi dengan harga diamonds sebagai target.

## Metrik

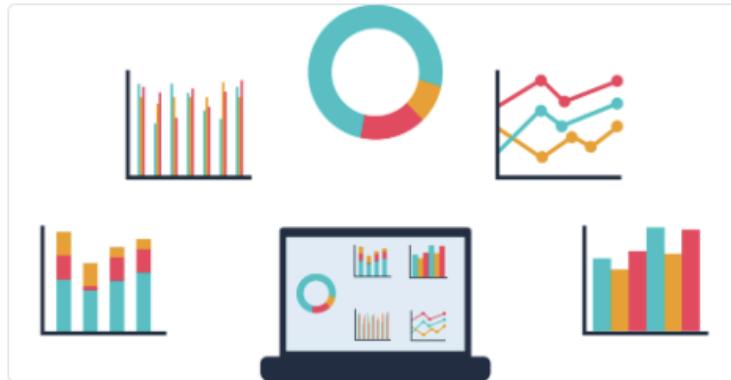
Metrik digunakan untuk mengevaluasi seberapa baik model Anda dalam memprediksi harga. Untuk kasus regresi, beberapa metrik yang biasanya digunakan adalah Mean Squared Error (MSE) atau Root Mean Square Error (RMSE). Secara umum, metrik ini mengukur seberapa jauh hasil prediksi dengan nilai yang sebenarnya. Kita akan bahas lebih detail mengenai metrik ini di modul Evaluasi.

Pengembangan model akan menggunakan beberapa algoritma machine learning yaitu K-Nearest Neighbor, Random Forest, dan Boosting Algorithm. Dari ketiga model ini, akan dipilih satu model yang memiliki nilai kesalahan prediksi terkecil. Dengan kata lain, kita akan membuat model seakurat mungkin, yaitu model dengan nilai kesalahan sekecil mungkin.

Membuat model prediktif dengan machine learning tentu memerlukan data. Berita baiknya adalah, perusahaan memiliki data yang dibutuhkan untuk membuat model prediksi. Dataset yang akan kita gunakan pada praktik kali ini adalah Diamond dataset. Ia merupakan *real-world* dataset yang menjadi data bawaan (*built in*) dari [ggplot2 packages](#). Anda dapat mengunduhnya melalui [repository GitHub ggplot](#).

Pada modul berikutnya, Anda akan mengeksplorasi dataset ini lebih detail. Sudah siap untuk lanjut?

## Data Understanding



Data yang Anda gunakan pada proyek kali ini adalah Diamond dataset yang diunduh dari [repository GitHub ggplot](#). Untuk tahap latihan, Anda hanya akan menggunakan dataset ini sehingga tidak perlu menambahkan data lain atau melakukan peng gabungan dataset lagi. Selain itu, dataset ini juga cukup bersih sehingga tidak terlalu banyak memerlukan proses data cleaning.

Dataset ini memiliki 53.940 jenis diamonds dengan berbagai karakteristik dan harga. Karakteristik yang dimaksud di sini adalah fitur non-numerik seperti *cut*, *color*, dan *clarity*, serta fitur numerik seperti *carat*, *x*, *y*, *z*, *table*, dan *depth*. Kesembilan fitur ini adalah fitur yang akan Anda gunakan dalam menemukan pola pada data, sedangkan harga merupakan fitur target.

Dalam membuat model prediktif dengan machine learning, Anda dapat menggunakan tools manapun yang biasa Anda gunakan, misalnya Google Colaboratory, Jupyter Notebook, Python Anaconda. Jika ingin menggunakan Python IDE (integrated Development Tools) Anda bisa menggunakan tools seperti Pycharm atau Visual Studio. Selain itu, Anda juga dapat menggunakan notebook dari penyedia layanan cloud seperti IBM Watson Studio.

Untuk kemudahan dalam menguraikan setiap tahapan kode, proyek ini menggunakan tools Google Colaboratory. Bagaimana dengan Anda? Sudah menentukan tools apakah yang ingin digunakan?

Jika sudah, mari langsung coba memahami data kita dalam beberapa tahapan berikut:

- Data loading
- Exploratory Data Analysis - Deskripsi Variabel
- Exploratory Data Analysis - Menangani Missing Value dan Outliers
- Exploratory Data Analysis - Univariate Analysis
- Exploratory Data Analysis - Multivariate Analysis

## Data Loading

Supaya isi dataset lebih mudah dipahami, kita perlu melakukan proses *loading data* terlebih dahulu. Jangan lupa import library pandas untuk dapat membaca file datanya. Di sini, kita membaca data langsung dari url sumber data di [repository GitHub ggplot](#). Dataset yang akan kita gunakan bernama diamonds.csv.

Selain cara di atas, terdapat beberapa cara lain untuk membaca dataset. Jika kita memiliki file dataset dalam komputer atau local machine, kita bisa upload dataset tersebut langsung ke *file storage* di Google Colab. Jika menggunakan tools lain atau mengunggah data ke Google Drive, pastikan Anda menyesuaikan path datanya ya.

Pertama, import library yang dibutuhkan. Anda dapat melakukannya di awal, atau di tiap kode sel.

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4. %matplotlib inline
5. import seaborn as sns
```

Kemudian, tuliskan kode berikut.

```
1. # load the dataset
2. url = 'https://raw.githubusercontent.com/tidyverse/ggplot2/master/data-raw/diamonds.csv'
3. diamonds = pd.read_csv(url)
4. diamonds
```

Hasilnya sebagai berikut.

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64
53940 rows × 10 columns										

Output kode di atas memberikan informasi sebagai berikut:

- Ada 53.940 baris (records atau jumlah pengamatan) dalam dataset.
- Terdapat 10 kolom yaitu: carat, cut, color, clarity, depth, table, price, x, y, z.

## Exploratory Data Analysis - Deskripsi Variabel

Exploratory data analysis atau sering disingkat EDA merupakan proses investigasi awal pada data untuk menganalisis karakteristik, menemukan pola, anomali, dan memeriksa asumsi pada data. Teknik ini biasanya menggunakan bantuan statistik dan representasi grafis atau visualisasi. Teknik ini awalnya dikembangkan oleh seorang matematikawan Amerika bernama John Tukey pada tahun 1970 [21]. Teknik EDA kemudian semakin berkembang dan digunakan secara luas dalam proses analisis data hingga saat ini.



Cakupan proses EDA sangat luas. Namun, secara umum, Anda dapat melakukan proses EDA untuk menjawab beberapa pertanyaan berikut:

- Apa saja jenis variabel pada dataset?
- Bagaimana distribusi variabel dalam dataset?
- Apakah ada missing value?
- Apakah ada fitur yang tidak berguna (redundant)?
- Bagaimana korelasi antara fitur dan target?

Dalam menjawab pertanyaan-pertanyaan di atas, Anda perlu melakukan beberapa hal pada data seperti mengeliminasi fitur, membuat fitur baru, menggabungkan kategori pada fitur non-numerik, melakukan transformasi pada fitur, dan lain-lain. Perlu diingat bahwa penerapan teknik EDA tentu berbeda antara satu data dan lainnya. Meskipun ada pedoman tentang teknik EDA mana yang berguna dalam situasi tertentu, EDA adalah proses seni. Keterampilan yang diperoleh melalui latihan, pengamatan, proses belajar dan pengalaman mempengaruhi proses EDA yang Anda lakukan. Selain itu, EDA tidak dibatasi oleh teknik tertentu sehingga Anda kadang perlu menemukan cara baru atau melakukan improvisasi dalam melihat data.

Umumnya, teknik EDA dibagi menjadi dua cara. Pertama, setiap metode bersifat grafis atau non-grafis. Kedua, setiap metode bersifat univariate (melibatkan satu variate atau variabel) dan multivariate (melibatkan dua atau lebih variabel).

Univariate berasal dari kata "uni" yang artinya satu dan "variate" yang berarti variasi. Jadi, analisis univariate adalah cara kita melakukan analisis terhadap satu jenis (variasi) variabel saja. Dengan kata lain, analisis univariate merupakan proses untuk mengeksplorasi dan menjelaskan setiap variabel dalam kumpulan data secara terpisah.

Sedangkan, multivariate berasal dari kata "multi" yang artinya banyak dan "variate" yang berarti variasi. Jadi, analisis multivariate adalah cara kita melakukan analisis terhadap banyak variasi variabel. Dengan kata lain, multivariate analysis merupakan proses eksplorasi yang melibatkan banyak (dua atau lebih) variabel pada data.

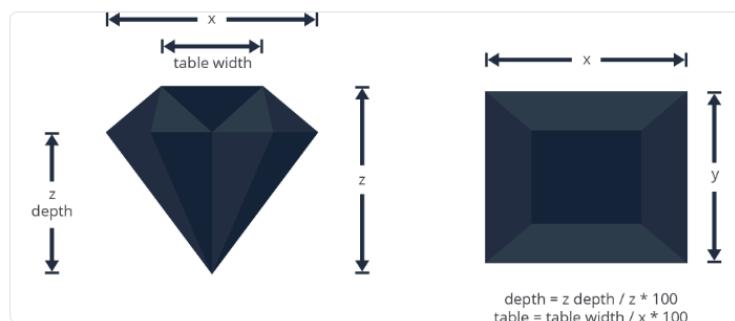
Untuk lebih jelasnya, yuk langsung kita terapkan proses EDA pada dataset!

## Deskripsi Variabel

Berdasarkan informasi dari [Kaggle](#), variabel-variabel pada Diamond dataset adalah sebagai berikut:

- Harga dalam dolar Amerika Serikat (\$) adalah fitur target.
- carat: merepresentasikan bobot (weight) dari diamonds (0.2-5.01), digunakan sebagai ukuran dari batu permata dan perhiasan.
- cut: merepresentasikan kualitas pemotongan diamonds (Fair, Good, Very Good, Premium, and Ideal).
- color: merepresentasikan warna, dari J (paling buruk) ke D (yang terbaik).
- clarity: merepresentasikan seberapa jernih diamonds (I1 (paling buruk), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (terbaik))
- x: merepresentasikan panjang diamonds dalam mm (0-10.74).
- y: merepresentasikan lebar diamonds dalam mm (0-58.9).
- z: merepresentasikan kedalaman diamonds dalam mm (0-31.8).
- depth: merepresentasikan  $z/\text{mean}(x, y) = 2 * z/(x + y)$  (43-79).
- table: merepresentasikan lebar bagian atas berlian relatif terhadap titik terlebar 43-95).

Berikut adalah ilustrasi mengenai fitur x, y, z, depth, dan table.



Setelah memahami deskripsi variabel pada data, langkah selanjutnya adalah mengecek informasi pada dataset dengan fungsi info() berikut.

```
1. diamonds.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   carat    53940 non-null   float64
 1   cut      53940 non-null   object 
 2   color    53940 non-null   object 
 3   clarity  53940 non-null   object 
 4   depth    53940 non-null   float64
 5   table    53940 non-null   float64
 6   price    53940 non-null   int64  
 7   x        53940 non-null   float64
 8   y        53940 non-null   float64
 9   z        53940 non-null   float64
dtypes: float64(6), int64(1), object(3)
memory usage: 4.1+ MB
```

Dari output terlihat bahwa:

- Terdapat 3 kolom dengan tipe object, yaitu: cut, color, dan clarity. Kolom ini merupakan categorical features (fitur non-numerik).
- Terdapat 6 kolom numerik dengan tipe data float64 yaitu: carat, depth, table, x, y, dan z. Ini merupakan fitur numerik yang merupakan hasil pengukuran secara fisik.
- Terdapat 1 kolom numerik dengan tipe data int64, yaitu: price. Kolom ini merupakan target fitur kita.

Uraian di atas menunjukkan bahwa setiap kolom telah memiliki tipe data yang sesuai. Selanjutnya, Anda perlu mengecek deskripsi statistik data dengan fitur describe().

```
1. diamonds.describe()
```

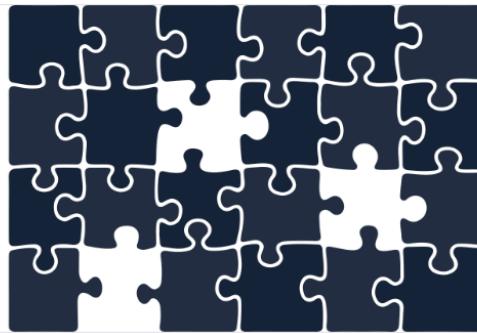
Output:

	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

Fungsi describe() memberikan informasi statistik pada masing-masing kolom, antara lain:

- Count adalah jumlah sampel pada data.
- Mean adalah nilai rata-rata.
- Std adalah standar deviasi.
- Min yaitu nilai minimum setiap kolom.
- 25% adalah kuartil pertama. Kuartil adalah nilai yang menandai batas interval dalam empat bagian sebaran yang sama.
- 50% adalah kuartil kedua, atau biasa juga disebut median (nilai tengah).
- 75% adalah kuartil ketiga.
- Max adalah nilai maksimum.

## Menangani Missing Value



Dari hasil fungsi describe(), nilai minimum untuk kolom x, y, dan z adalah 0. Seperti kita tahu, x, y, dan z adalah ukuran panjang, lebar, dan kedalaman diamonds sehingga tidak mungkin ada diamonds dengan dimensi x, y, atau z bernilai 0. Kita patut menduga bahwa ini merupakan data yang tidak valid atau sering disebut missing value. Mari kita cek ada berapa missing value pada kolom x, y, dan z.

```
1. x = (diamonds.x == 0).sum()
2. y = (diamonds.y == 0).sum()
3. z = (diamonds.z == 0).sum()
4.
5. print("Nilai 0 di kolom x ada: ", x)
6. print("Nilai 0 di kolom y ada: ", y)
7. print("Nilai 0 di kolom z ada: ", z)
```

Output:

```
Nilai 0 di kolom x ada: 8
```

```
Nilai 0 di kolom y ada: 7
```

```
Nilai 0 di kolom z ada: 20
```

Selanjutnya, mari kita cek, apakah data bernilai 0 pada salah satu dimensi juga terdapat pada dimensi lain? Kita cek dari kolom z yang memiliki jumlah missing value terbanyak.

```
1. diamonds.loc[(diamonds['z']==0)]
```

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
2207	2208	1.00	Premium	G	SI2	59.1	59.0	3142	6.55	6.48	0.0
2314	2315	1.01	Premium	H	I1	58.1	59.0	3157	6.66	6.60	0.0
4791	4792	1.10	Premium	G	SI2	63.0	59.0	3696	6.50	6.47	0.0
5471	5472	1.01	Premium	F	SI2	59.2	58.0	3837	6.50	6.47	0.0
10167	10168	1.50	Good	G	I1	64.0	61.0	4731	7.15	7.04	0.0
11182	11183	1.07	Ideal	F	SI2	61.6	56.0	4954	0.00	6.62	0.0
11963	11964	1.00	Very Good	H	VS2	63.3	53.0	5139	0.00	0.00	0.0
13601	13602	1.15	Ideal	G	VS2	59.2	56.0	5564	6.88	6.83	0.0
15951	15952	1.14	Fair	G	VS1	57.5	67.0	6381	0.00	0.00	0.0
24394	24395	2.18	Premium	H	SI2	59.4	61.0	12631	8.49	8.45	0.0
24520	24521	1.56	Ideal	G	VS2	62.2	54.0	12800	0.00	0.00	0.0
26123	26124	2.25	Premium	I	SI1	61.3	58.0	15397	8.52	8.42	0.0
26243	26244	1.20	Premium	D	VVS1	62.1	59.0	15686	0.00	0.00	0.0
27112	27113	2.20	Premium	H	SI1	61.2	59.0	17265	8.42	8.37	0.0
27429	27430	2.25	Premium	H	SI2	62.8	59.0	18034	0.00	0.00	0.0
27503	27504	2.02	Premium	H	VS2	62.7	53.0	18207	8.02	7.95	0.0
27739	27740	2.80	Good	G	SI2	63.8	58.0	18784	8.90	8.85	0.0
49556	49557	0.71	Good	F	SI2	64.1	60.0	2130	0.00	0.00	0.0
49557	49558	0.71	Good	F	SI2	64.1	60.0	2130	0.00	0.00	0.0
51506	51507	1.12	Premium	G	I1	60.4	59.0	2383	6.71	6.67	0.0

Menarik! Ternyata, seluruh data bernilai 0 pada dimensi x dan y juga memiliki nilai 0 pada dimensi z. Sekarang, kita perlu memutuskan apa yang akan kita lakukan terhadap data yang hilang ini. Ada beberapa teknik untuk mengatasi missing value, antara lain, menghapus atau melakukan drop terhadap data yang hilang, menggantinya dengan mean atau median, serta memprediksi dan mengganti nilainya dengan teknik regresi [22].

Setiap teknik tentu memiliki kelebihan dan kekurangan. Selain itu, penanganan missing value juga bersifat unik tergantung kasusnya. Pada kasus kita, 20 sampel missing value merupakan jumlah yang kecil jika dibandingkan dengan jumlah total sampel yaitu 53.940. Jika 20 sampel ini dihapus, tentu kita akan kehilangan beberapa informasi. Akan tetapi, ini tidak jadi masalah sebab kita masih memiliki 53.920 sampel lainnya. Oleh karena itu, mari kita hapus saja missing value ini.

```
1. # Drop baris dengan nilai 'x', 'y', dan 'z' = 0
2. diamonds = diamonds.loc[(diamonds[['x','y','z']]!=0).all(axis=1)]
3.
4. # Cek ukuran data untuk memastikan baris sudah di-drop
5. diamonds.shape
```

Output:

```
(53920, 10)
```

Setelah baris bernilai 0 dihapus, jumlah sampel atau baris data berubah menjadi 53.920. Mari kita cek lagi dengan fungsi describe() untuk memastikan tidak ada nilai 0 lagi pada kolom x, y, dan z.

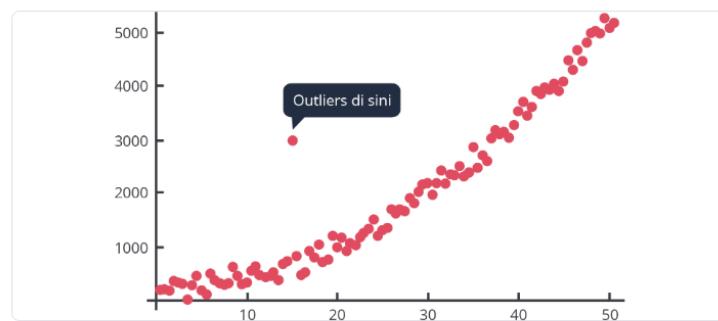
```
diamonds.describe()
```

	carat	depth	table	price	x	y	z
count	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000
mean	0.797688	61.749514	57.456834	3930.993231	5.731627	5.754887	3.540046
std	0.473795	1.432331	2.234064	3987.280446	1.119423	1.140126	0.702530
min	0.200000	43.000000	43.000000	326.000000	3.730000	3.680000	1.070000
25%	0.400000	61.000000	56.000000	949.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5323.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

Nilai minimum pada kolom x, y, dan z sudah bukan 0 lagi. Kita bisa lanjutkan ke tahapan selanjutnya yaitu menangani outliers.

## Menangani Outliers

Beberapa pengamatan dalam satu set data kadang berada di luar lingkungan pengamatan lainnya. Pengamatan seperti itu disebut outlier. Menurut Kuhn dan Johnson dalam Applied Predictive Modeling [23], outliers adalah sampel yang nilainya sangat jauh dari cakupan umum data utama. Ia adalah hasil pengamatan yang kemunculannya sangat jarang dan berbeda dari data hasil pengamatan lainnya.



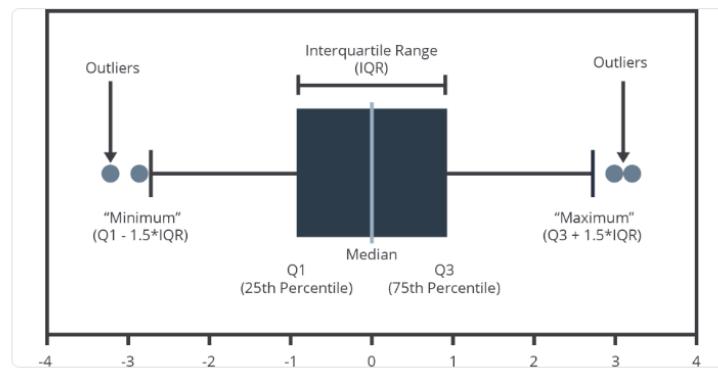
Ada beberapa teknik untuk menangani outliers, antara lain:

- Hypothesis Testing
- Z-score method
- IQR Method

Pada kasus ini, Anda akan mendeteksi outliers dengan teknik visualisasi data (boxplot). Kemudian, Anda akan menangani outliers dengan teknik IQR method. IQR adalah singkatan dari Inter Quartile Range. Untuk memahami apa itu IQR, mari kita ingat lagi konsep kuartil. Kuartil dari suatu populasi adalah tiga nilai yang membagi distribusi data menjadi empat sebaran. Seperempat dari data berada di bawah kuartil pertama (Q1), setengah dari data berada di bawah kuartil kedua (Q2), dan tiga perempat dari data berada di kuartil ketiga (Q3). Dengan demikian interquartile range atau  $IQR = Q_3 - Q_1$ .

Sampai di sini, mari kita simpan dulu metode IQR. Anda akan menggunakannya nanti jika memang terbukti ada outliers pada dataset kita. Untuk mengeceknya, kita akan menggunakan teknik visualisasi, yaitu jenis boxplot. Menurut Seltman dalam "Experimental Design and Analysis" [24], boxplot menunjukkan ukuran lokasi dan penyebaran, serta memberikan informasi tentang simetri dan outliers. Boxplot bisa digambarkan secara vertikal maupun horizontal.

Berikut adalah ilustrasi dan penjelasan nilai statistik pada boxplot.



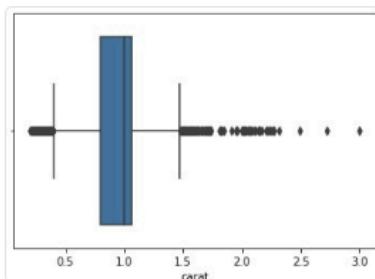
Sekarang, mari kita visualisasikan data Diamonds dengan boxplot untuk mendeteksi outliers pada beberapa fitur numerik. Anda dapat melakukan visualisasi pada fitur numerik sisanya.

Sekarang, mari kita visualisasikan data Diamonds dengan boxplot untuk mendeteksi outliers pada beberapa fitur numerik. Anda dapat melakukan visualisasi pada fitur numerik sisanya.

#### 1. Carat

```
1. sns.boxplot(x=diamonds['carat'])
```

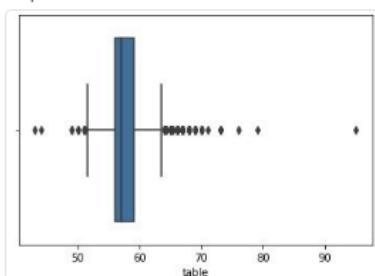
Output:



#### 2. Fitur Table

```
1. sns.boxplot(x=diamonds['table'])
```

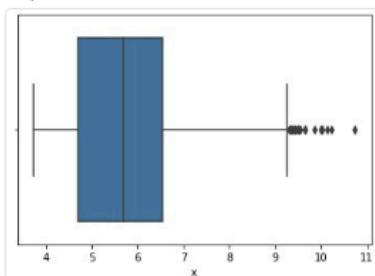
Output:



#### 3. Fitur x

```
1. sns.boxplot(x=diamonds['x'])
```

Output:



Jika kita perhatikan kembali, pada beberapa fitur numerik di atas terdapat outliers. Tugas Anda selanjutnya adalah mengatasi outliers tersebut dengan metode yang telah kita bahas sebelumnya yaitu metode IQR. Anda akan menggunakan metode IQR untuk mengidentifikasi outlier yang berada di luar Q1 dan Q3. Nilai apa pun yang berada di luar batas ini dianggap sebagai outlier.

Seltman dalam "Experimental Design and Analysis" [24] menyatakan bahwa outliers yang diidentifikasi oleh boxplot (disebut juga "boxplot outliers") didefinisikan sebagai data yang nilainya 1.5 QR di atas Q3 atau 1.5 QR di bawah Q1. Jangan bingung dulu ya, kita akan bahas lebih detail.

Hal pertama yang perlu Anda lakukan adalah membuat batas bawah dan batas atas. Untuk membuat batas bawah, kurangi Q1 dengan  $1.5 * \text{IQR}$ . Kemudian, untuk membuat batas atas, tambahkan  $1.5 * \text{IQR}$  dengan Q3. Mudah bukan

Berikut persamaannya:

$$\text{Batas bawah} = Q1 - 1.5 * IQR$$

$$\text{Batas atas} = Q3 + 1.5 * IQR$$

Mari kita terapkan persamaan ini ke dalam code.

```
1. Q1 = diamonds.quantile(0.25)
2. Q3 = diamonds.quantile(0.75)
3. IQR=Q3-Q1
4. diamonds=diamonds[~((diamonds<(Q1-1.5*IQR))|(diamonds>(Q3+1.5*IQR))).any(axis=1)]
5.
6. # Cek ukuran dataset setelah kita drop outliers
7. diamonds.shape
```

Output:

```
(47524, 10)
```

Dataset Anda sekarang telah bersih dan memiliki 47.524 sampel.

## Univariate Analysis

Selanjutnya, kita akan melakukan proses analisis data dengan teknik Univariate EDA. Pertama, Anda bagi fitur pada dataset menjadi dua bagian, yaitu numerical features dan categorical features.

```
1. numerical_features = ['price', 'carat', 'depth', 'table', 'x', 'y', 'z']
2. categorical_features = ['cut', 'color', 'clarity']
```

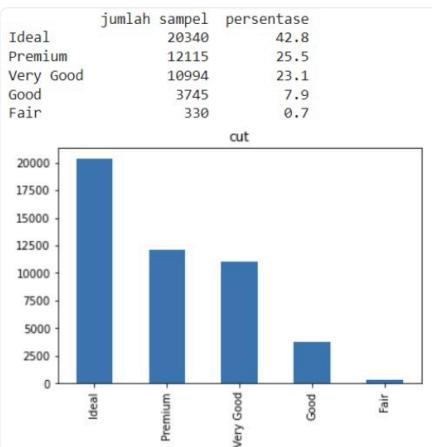
Lakukan analisis terhadap fitur kategori terlebih dahulu.

## Categorical Features

### Fitur Cut

```
1. feature = categorical_features[0]
2. count = diamonds[feature].value_counts()
3. percent = 100*diamonds[feature].value_counts(normalize=True)
4. df = pd.DataFrame({'jumlah sampel':count, 'persentase':percent.round(1)})
5. print(df)
6. count.plot(kind='bar', title=feature);
```

Output:



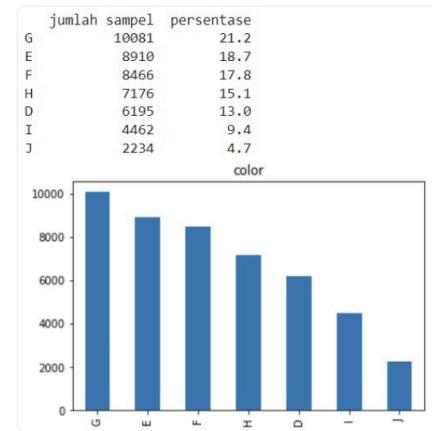
Terdapat 5 kategori pada fitur Cut, secara berurutan dari jumlahnya yang paling banyak yaitu: Ideal, Premium, Very Good, Good, dan Fair. Dari data persentase dapat kita simpulkan bahwa lebih dari 60% sampel merupakan diamonds tipe grade tinggi, yaitu grade Ideal dan Premium.

## Fitur Color

Selanjutnya, mari kita cek fitur 'Color'.

```
1. feature = categorical_features[1]
2. count = diamonds[feature].value_counts()
3. percent = 100*diamonds[feature].value_counts(normalize=True)
4. df = pd.DataFrame({'jumlah sampel':count, 'persentase':percent.round(1)})
5. print(df)
6. count.plot(kind='bar', title=feature);
```

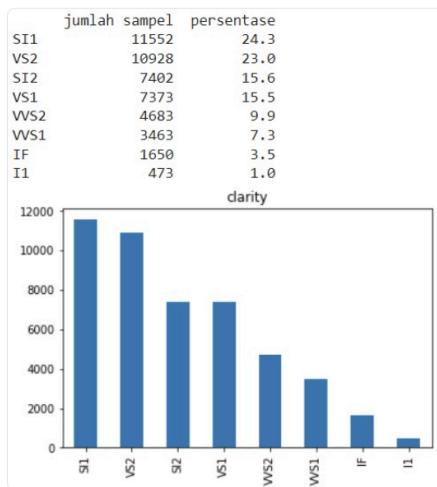
Output:



Berdasarkan deskripsi variabel, urutan kategori warna dari yang paling buruk ke yang paling bagus adalah J, I, H, G, F, E, dan D. Dari grafik di atas, dapat kita simpulkan bahwa sebagian besar grade berada pada grade menengah, yaitu G, F, H.

## Fitur Clarity

Dengan code yang sama untuk fitur clarity, hasil plotnya adalah sebagai berikut:



Berdasarkan informasi dari deskripsi variabel, fitur Clarity terdiri dari 8 kategori dari yang paling buruk ke yang paling baik, yaitu: I1, SI2, SI1, VS2, VS1, VVS2, VVS1, dan IF.

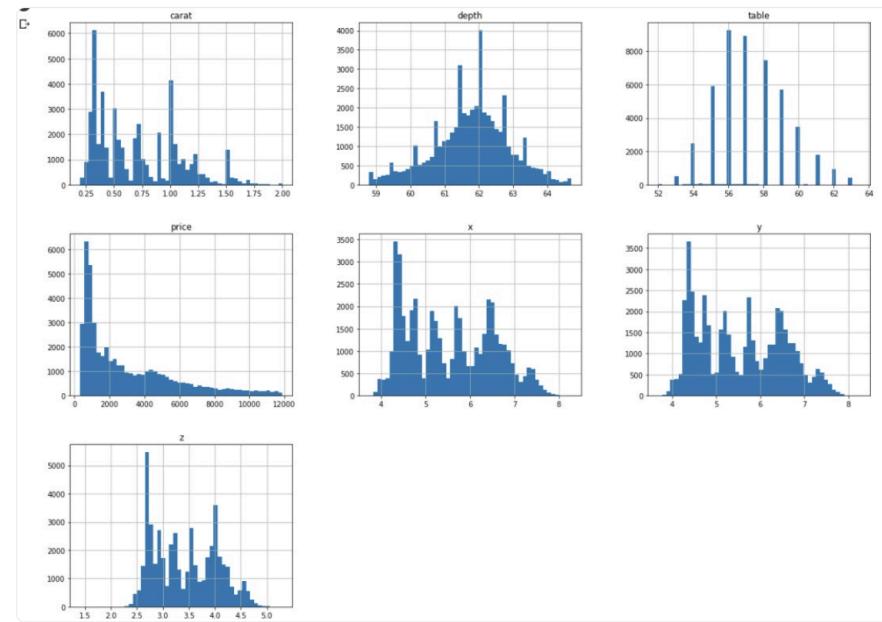
1. 'IF' - Internally Flawless
2. 'VVS2' - Very Very Slight Inclusions
3. 'VVS1' - Very Very Slight Inclusions
4. 'VS1' - Very Slight Inclusions
5. 'VS2' - Very Slight Inclusions
6. 'SI2' - Slight Inclusions
7. 'SI1' - Slight Inclusions
8. 'I1' - Imperfect

Dari grafik kita bisa menyimpulkan bahwa sebagian besar fitur merupakan grade rendah, yaitu SI1, SI2, dan VS2.

## Numerical Features

Selanjutnya, untuk fitur numerik, kita akan melihat histogram masing-masing fiturnya menggunakan code berikut.

```
1. diamonds.hist(bins=50, figsize=(20,15))  
2. plt.show()
```



Mari amati histogram di atas, khususnya histogram untuk variabel "price" yang merupakan fitur target (label) pada data kita. Dari histogram "price", kita bisa memperoleh beberapa informasi, antara lain:

- Peningkatan harga diamonds sebanding dengan penurunan jumlah sampel. Hal ini dapat kita lihat jelas dari histogram "price" yang grafiknya mengalami penurunan seiring dengan semakin banyaknya jumlah sampel (sumbu y).
- Rentang harga diamonds cukup tinggi yaitu dari skala ratusan dolar Amerika hingga sekitar \$11800.
- Setengah harga berlian bernilai di bawah \$2500.
- Distribusi harga miring ke kanan (right-skewed). Hal ini akan berimplikasi pada model.

## Exploratory Data Analysis - Multivariate Analysis

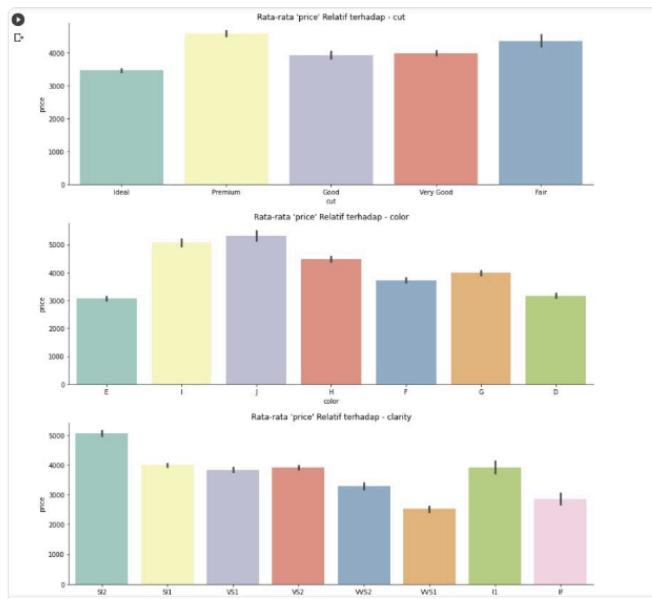
Multivariate EDA menunjukkan hubungan antara dua atau lebih variabel pada data. Multivariate EDA yang menunjukkan hubungan antara dua variabel biasa disebut sebagai bivariate EDA. Selanjutnya, kita akan melakukan analisis data pada fitur kategori dan numerik.

### Categorical Features

Pada tahap ini, kita akan mengecek rata-rata harga terhadap masing-masing fitur untuk mengetahui pengaruh fitur kategori terhadap harga.

```
1. cat_features = diamonds.select_dtypes(include='object').columns.to_list()
2.
3. for col in cat_features:
4.     sns.catplot(x=col, y="price", kind="bar", dodge=False, height = 4, aspect = 3, data=diamonds)
5.     plt.title("Rata-rata 'price' Relatif terhadap - {}".format(col))
```

Output:



Dengan mengamati rata-rata harga relatif terhadap fitur kategori di atas, kita memperoleh *insight* sebagai berikut:

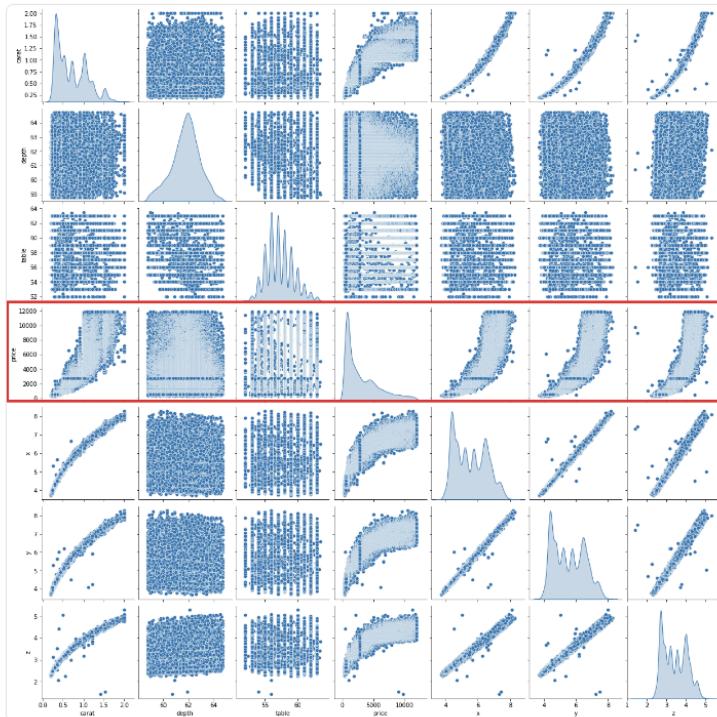
- Pada fitur 'cut', rata-rata harga cenderung mirip. Rentangnya berada antara 3500 hingga 4500. Grade tertinggi yaitu grade Ideal memiliki harga rata-rata terendah diantara grade lainnya. Sehingga, fitur cut memiliki pengaruh atau dampak yang kecil terhadap rata-rata harga.
- Pada fitur 'color', semakin rendah grade warna, harga diamonds justru semakin tinggi. Dari sini dapat disimpulkan bahwa warna memiliki pengaruh yang rendah terhadap harga.
- Pada fitur 'clarity', secara umum, diamond dengan grade lebih rendah memiliki harga yang lebih tinggi. Hal ini berarti bahwa fitur 'clarity' memiliki pengaruh yang rendah terhadap harga.
- Kesimpulan akhir, fitur kategori memiliki pengaruh yang rendah terhadap harga.

## Numerical Features

Untuk mengamati hubungan antara fitur numerik, kita akan menggunakan fungsi `pairplot()`. Kita juga akan mengobservasi korelasi antara fitur numerik dengan fitur target menggunakan fungsi `corr()`. Tidak perlu menunggu lama, mari kita langsung analisis datanya.

```
1. # Mengamati hubungan antar fitur numerik dengan fungsi pairplot()
2. sns.pairplot(diamonds, diag_kind = 'kde')
```

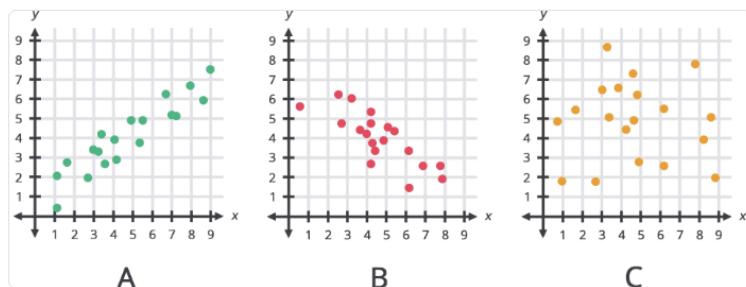
Output:



Fungsi `pairplot` dari library `seaborn` menunjukkan relasi pasangan dalam dataset. Misalnya, relasi antara fitur 'x' pada sumbu x dengan fitur 'z' pada sumbu y atau relasi antara fitur 'carat' pada sumbu x dengan 'price' pada sumbu y. Dari grafik, kita dapat melihat plot relasi masing-masing fitur numerik pada dataset.

Pada kasus ini, kita akan melihat relasi antara semua fitur numerik dengan fitur target kita yaitu 'price'. Untuk membacanya, perhatikan fitur pada sumbu y, temukan fitur target 'price', dan lihatlah grafik relasi antara semua fitur pada sumbu x dengan fitur price pada sumbu y. Dalam hal ini, fitur 'price' berada pada baris keempat (dari atas) sumbu y (ditandai oleh kotak merah). Sehingga, kita cukup melihat relasi antar fitur numerik dengan fitur target 'price' pada baris tersebut saja.

Sebelum memperhatikan pola sebaran data pada grafik pairplot di atas, mari kita pahami terlebih dahulu cara membaca korelasi pada sebaran data. Korelasi pada fitur tampak dari adanya pola pada sebaran data. Sebaran data acak merupakan indikasi korelasi yang lemah (atau tidak ada korelasi sama sekali). Sedangkan, sebaran yang memiliki pola (tidak acak) merupakan indikasi adanya korelasi. Perhatikan contoh gambar berikut.



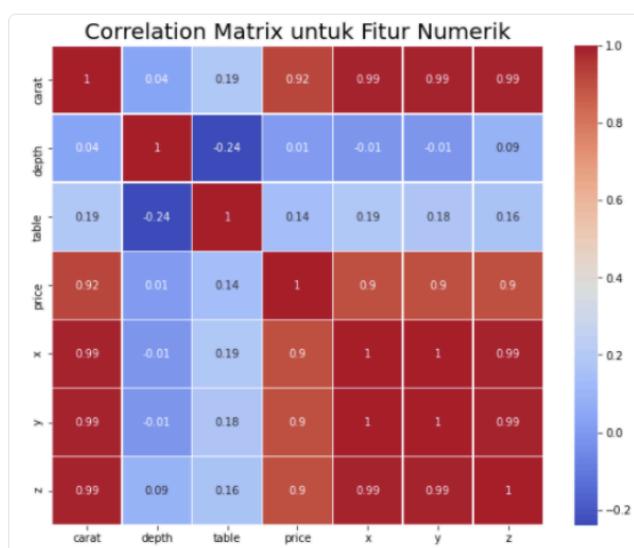
Dari pola sebaran data (titik-titik) pada gambar di atas, pola data grafik A memiliki korelasi positif. Hal ini ditandai dengan meningkatnya variabel pada sumbu y saat terjadi peningkatan variabel pada sumbu x. Sedangkan, pola data grafik B memiliki korelasi negatif yang ditandai dengan menurunnya variabel y saat terjadi kenaikan pada variabel x. Terakhir, sebaran pada data grafik C menunjukkan pola acak, artinya tidak ada korelasi data.

Pada pola sebaran data grafik pairplot sebelumnya, terlihat 'carat', 'x', 'y', dan 'z' memiliki korelasi yang tinggi dengan fitur "price". Sedangkan kedua fitur lainnya yaitu 'depth' dan 'table' terlihat memiliki korelasi yang lemah karena sebarannya tidak membentuk pola. Untuk mengevaluasi skor korelasinya, gunakan fungsi corr().

```

1. plt.figure(figsize=(10, 8))
2. correlation_matrix = diamonds[numerical_features].corr().round(2)
3.
4. # Untuk menge-print nilai di dalam kotak, gunakan parameter annot=True
5. sns.heatmap(data=correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5, )
6. plt.title("Correlation Matrix untuk Fitur Numerik ", size=20)

```



Sebelum mengamati gambar di atas, mari kita simak penjelasan mengenai hubungan korelasi antar fitur.

Koefisien korelasi berkisar antara -1 dan +1. Ia mengukur kekuatan hubungan antara dua variabel serta arahnya (positif atau negatif). Mengenai kekuatan hubungan antar variabel, semakin dekat nilainya ke 1 atau -1, korelasinya semakin kuat. Sedangkan, semakin dekat nilainya ke 0, korelasinya semakin lemah.

Arah korelasi antara dua variabel bisa bernilai positif (nilai kedua variabel cenderung meningkat bersama-sama) maupun negatif (nilai salah satu variabel cenderung meningkat ketika nilai variabel lainnya menurun).

Nah, kembali pada grafik korelasi di atas. Jika kita amati, fitur 'carat', 'x', 'y', dan 'z' memiliki skor korelasi yang besar (di atas 0.9) dengan fitur target 'price'. Artinya, fitur 'price' berkorelasi tinggi dengan keempat fitur tersebut. Sementara itu, fitur 'depth' memiliki korelasi yang sangat kecil (0.01). Sehingga, fitur tersebut dapat di-drop.

```
1. diamonds.drop(['depth'], inplace=True, axis=1)
2. diamonds.head()
```

Output:

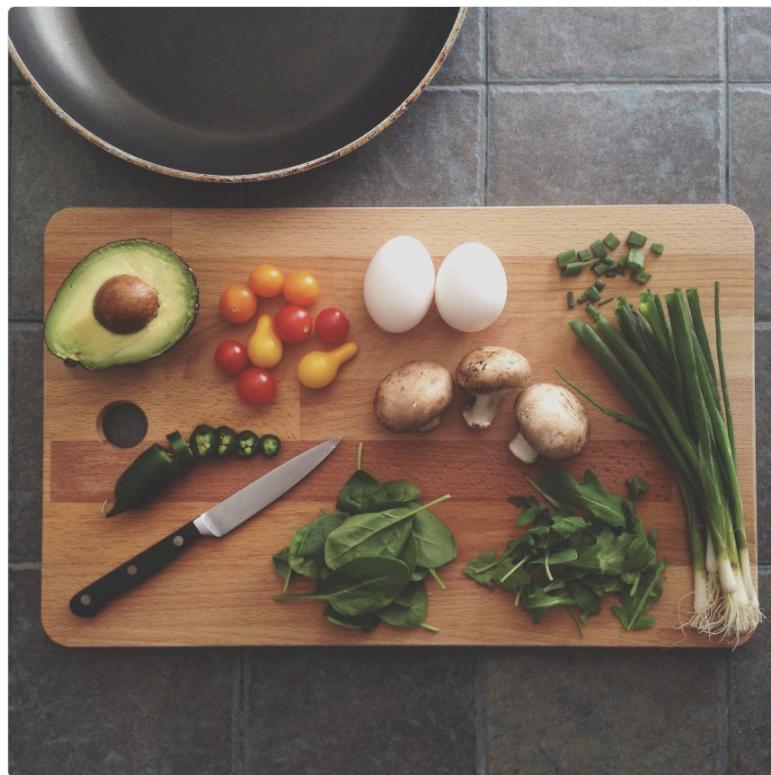
	carat	cut	color	clarity	table	price	x	y	z
0	0.23	Ideal	E	SI2	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	61.0	326	3.89	3.84	2.31
3	0.29	Premium	I	VS2	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	58.0	335	4.34	4.35	2.75
5	0.24	Very Good	J	VVS2	57.0	336	3.94	3.96	2.48

Sampai di sini kita telah melakukan analisis data dengan teknik EDA. Berikutnya, kita akan melakukan proses data preparation sebelum membuat model prediktif dengan machine learning. Tetap semangat ya!

---

## Data Preparation

Bayangkan Anda akan memasak. Sebelum memulai, tentunya Anda perlu melakukan persiapan terlebih dahulu. Tahap ini meliputi proses mencuci dan menakar bahan makanan, memotong-motong, serta membagi bahan mana yang akan dimasukkan terlebih dahulu dan bahan mana yang dimasukkan belakangan. Selain itu, kadang Anda juga melakukan pra-pemrosesan bahan makanan, misalnya merebus telur atau daging terlebih dahulu sebelum dimasak bersama bahan-bahan lainnya.



Demikian juga saat bekerja dengan data. Data preparation merupakan tahapan penting dalam proses pengembangan model machine learning. Ini adalah tahap di mana kita melakukan proses transformasi pada data sehingga menjadi bentuk yang cocok untuk proses pemodelan. Ada beberapa tahapan yang umum dilakukan pada data preparation, antara lain, seleksi fitur, transformasi data, *feature engineering*, dan *dimensionality reduction*.

Pada bagian ini kita akan melakukan empat tahap persiapan data, yaitu:

- Encoding fitur kategori.
- Reduksi dimensi dengan Principal Component Analysis (PCA).
- Pembagian dataset dengan fungsi `train_test_split` dari library `sklearn`.
- Standarisasi.

Tanpa menunggu lama, yuk langsung kita terapkan proses ini pada dataset.

## Encoding Fitur Kategori

Untuk melakukan proses encoding fitur kategori, salah satu teknik yang umum dilakukan adalah teknik *one-hot-encoding*. Library [scikit-learn](#) menyediakan fungsi ini untuk mendapatkan fitur baru yang sesuai sehingga dapat mewakili variabel kategori. Kita memiliki tiga variabel kategori dalam dataset kita, yaitu 'cut', 'color', dan 'clarity'. Mari kita lakukan proses encoding ini dengan fitur `get_dummies`.

```
1. from sklearn.preprocessing import OneHotEncoder
2. diamonds = pd.concat([diamonds, pd.get_dummies(diamonds['cut'], prefix='cut')],axis=1)
3. diamonds = pd.concat([diamonds, pd.get_dummies(diamonds['color'], prefix='color')],axis=1)
4. diamonds = pd.concat([diamonds, pd.get_dummies(diamonds['clarity'], prefix='clarity')],axis=1)
5. diamonds.drop(['cut','color','clarity'], axis=1, inplace=True)
6. diamonds.head()
```

Output:

	carat	table	price	x	y	z	cut_Fair	cut_Good	cut_Ideal	cut_Premium	cut_VeryGood	color_D	color_E	color_F	color_G	color_H	color_I	color_J	clarity_SI	clarity_VS1	clarity_VS2	clarity_SI1	clarity_VSI	clarity_VVS1	clarity_VVS2	clarity_VVS3	clarity_VVS4
0	0.22	56.0	326	3.99	3.98	2.42	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
1	0.21	61.0	326	3.89	3.84	2.31	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	
2	0.29	60.0	334	4.20	4.23	2.63	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	
3	0.31	60.0	336	4.34	4.36	2.75	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	
4	0.24	67.0	336	3.94	3.96	2.48	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	

Sekarang, variabel kategori kita telah berubah menjadi variabel numerik.

## Reduksi Dimensi dengan PCA

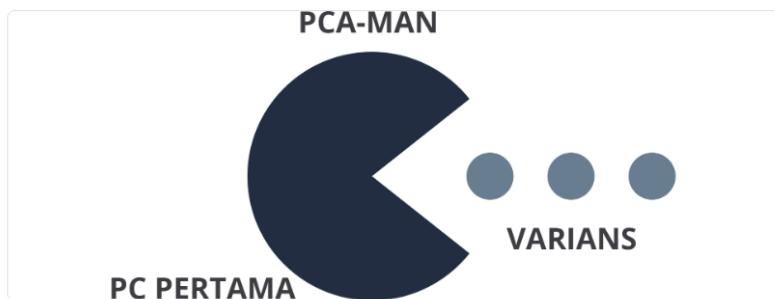
Teknik reduksi (pengurangan) dimensi adalah prosedur yang mengurangi jumlah fitur dengan tetap mempertahankan informasi pada data. Teknik pengurangan dimensi yang paling populer adalah Principal Component Analysis atau disingkat menjadi PCA. Ia adalah teknik untuk mereduksi dimensi, mengekstraksi fitur, dan mentransformasi data dari "n-dimensional space" ke dalam sistem berkoordinat baru dengan dimensi m, di mana m lebih kecil dari n.

PCA bekerja menggunakan metode aljabar linier. Ia mengasumsikan bahwa sekumpulan data pada arah dengan varians terbesar merupakan yang paling penting (utama). PCA umumnya digunakan ketika variabel dalam data memiliki korelasi yang tinggi. Korelasi tinggi ini menunjukkan data yang berulang atau *redundant*.

Karena hal inilah, teknik PCA digunakan untuk mereduksi variabel asli menjadi sejumlah kecil variabel baru yang tidak berkorelasi linier, disebut komponen utama (PC). Komponen utama ini dapat menangkap sebagian besar varians dalam variabel asli. Sehingga, saat teknik PCA diterapkan pada data, ia hanya akan menggunakan komponen utama dan mengabaikan sisanya.

Berikut penjelasan untuk masing-masing komponen utama (PC):

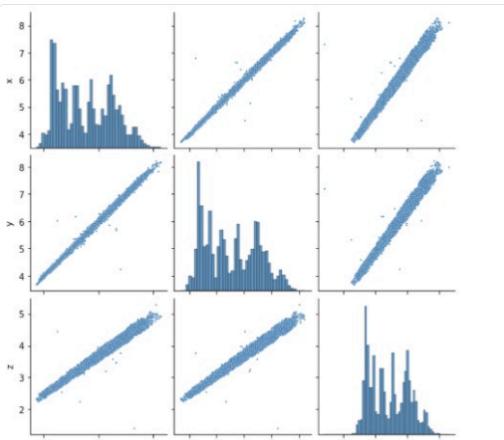
- PC pertama mewakili arah varians maksimum dalam data. Ia paling banyak menangkap informasi dari semua fitur dalam data.
- PC kedua menangkap sebagian besar informasi yang tersisa setelah PC pertama.
- PC ketiga menangkap sebagian besar informasi yang tersisa setelah PC pertama, PC kedua, dst.



Jika kita cek menggunakan fungsi pairplot, ketiga fitur ukuran diamonds dalam kolom 'x', 'y', dan 'z' memiliki korelasi yang tinggi. Hal ini karena ketiga fitur ini memiliki informasi yang sama, yaitu ukuran diamonds.

```
1. sns.pairplot(diamonds[['x','y','z']], plot_kws={"s": 3});
```

Output:



Selanjutnya, aplikasikan class **PCA** dari library scikit learn dengan kode berikut.

```
1. from sklearn.decomposition import PCA  
2.  
3. pca = PCA(n_components=3, random_state=123)  
4. pca.fit(diamonds[['x','y','z']])  
5. princ_comp = pca.transform(diamonds[['x','y','z']])
```

Kode di atas memanggil class **PCA()** dari library scikit-learn. Parameter yang kita masukkan ke dalam class adalah **n\_components** dan **random\_state**. Parameter **n\_components** merupakan jumlah komponen atau dimensi, dalam kasus kita jumlahnya ada 3, yaitu 'x', 'y', dan 'z'.

Sedangkan, parameter **random\_state** berfungsi untuk mengontrol random number generator yang digunakan. Parameter ini berupa bilangan integer dan nilainya bebas. Pada kasus ini, kita menerapkan **random\_state = 123**. Berapa pun nilai integer yang kita tentukan --selama itu bilangan integer, ia akan memberikan hasil yang sama setiap kali dilakukan pemanggilan fungsi (dalam kasus kita, class **PCA**).

Menentukan parameter **random\_state** bertujuan untuk dapat memastikan bahwa hasil pembagian dataset konsisten dan memberikan data yang sama setiap kali model dijalankan. Jika tidak ditentukan, maka tiap kali melakukan split, kita akan mendapatkan data train dan tes berbeda. Hal ini berpengaruh terhadap akurasi model ML yang menjadi berbeda tiap kali di-*run*.

Nah, setelah menerapkan class **PCA**, kita bisa mengetahui proporsi informasi dari ketiga komponen tadi.

```
1. pca.explained_variance_ratio_.round(3)
```

Output:

```
array([0.998, 0.002, 0.001])
```

Arti dari output di atas adalah, 99.8% informasi pada ketiga fitur 'x', 'y', 'z' terdapat pada PC pertama. Sedangkan sisanya, sebesar 0.2% dan 0.1% terdapat pada PC kedua dan ketiga. Perhatikanlah, jumlahnya jadi > 100%. Hal ini disebabkan oleh proses pembulatan (**round**) dalam 3 desimal ya, jadi tidak perlu khawatir.

Berdasarkan hasil ini, kita akan mereduksi fitur (dimensi) dan hanya mempertahankan PC (komponen) pertama saja. PC pertama ini akan menjadi fitur dimensi atau ukuran berlian menggantikan ketiga fitur lainnya ('x', 'y', 'z'). Kita beri nama fitur ini 'dimension'.

Sekarang Anda akan membuat fitur baru bernama 'dimension' untuk menggantikan fitur 'x', 'y', dan 'z'. Oleh karena itu, mari jalankan kode di atas dengan beberapa perubahan berikut:

- Gunakan n\_component = 1, karena kali ini, jumlah komponen kita hanya satu.
- Fit model dengan data masukan.
- Tambahkan fitur baru ke dataset dengan nama 'dimension' dan lakukan proses transformasi.
- Drop kolom 'x', 'y', dan 'z'.

```
1. from sklearn.decomposition import PCA
2. pca = PCA(n_components=1, random_state=123)
3. pca.fit(diamonds[['x','y','z']])
4. diamonds['dimension'] = pca.transform(diamonds.loc[:, ('x','y','z')]).flatten()
5. diamonds.drop(['x','y','z'], axis=1, inplace=True)
```

Output:

	carat	table	price	cut_Fair	cut_Good	cut_Ideal	cut_Premium	cut_Very Good	color_D	color_E	color_F	color_G	color_H	color_I	color_J	clarity_I1	clarity_I2	clarity_SI1	clarity_SI2	clarity_VS1	clarity_VS2	clarity_VV1	clarity_VV2	dimension
0	0.23	55.0	326	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0.245268	
1	0.21	61.0	326	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0.263006	
2	0.29	58.0	334	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0.248530	
3	0.31	58.0	335	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	-0.181197	
4	0.24	57.0	336	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	-0.245192	

## Train-Test-Split

Membagi dataset menjadi data latih (train) dan data uji (test) merupakan hal yang harus kita lakukan sebelum membuat model. Kita perlu mempertahankan sebagian data yang ada untuk menguji seberapa baik generalisasi model terhadap data baru. Ketahuilah bahwa setiap transformasi yang kita lakukan pada data juga merupakan bagian dari model. Karena data uji (test set) berperan sebagai data baru, kita perlu melakukan semua proses transformasi dalam data latih. Inilah alasan mengapa langkah awal adalah membagi dataset sebelum melakukan transformasi apa pun [25]. Tujuannya adalah agar kita tidak mengotori data uji dengan informasi yang kita dapat dari data latih.

Selama ini, kesalahan yang sering dilakukan oleh praktisi machine learning adalah melakukan proses scaling (penyekalaan) seperti normalisasi dan standarisasi sebelum membagi data menjadi data latih dan data uji. Hal ini berpotensi menimbulkan kebocoran data (data leakage). Proses scaling pada seluruh dataset membuat model memiliki informasi mengenai distribusi pada data uji. Informasi tentang data uji (yang seharusnya tidak dilihat oleh model) turut diikutsertakan dalam proses transformasi data latih. Oleh karena itu, kita akan melakukan proses scaling secara terpisah antara data latih dan data uji.

Kembali pada bahasan mengenai train\_test\_split, proporsi pembagian data latih dan uji biasanya adalah 80:20. Ingatlah bahwa proporsi ini hanya kebiasaan umum saja. Tujuan dari data uji adalah untuk untuk mengukur kinerja model pada data baru. Jadi, jika dataset yang kita miliki berukuran sangat kecil, misalnya kurang dari 1.000 sampel, maka pembagian 80:20 ini cukup ideal. Namun, jika memiliki dataset berukuran besar, kita perlu memikirkan strategi pembagian dataset lain agar proporsi data uji tidak terlalu banyak.

Sebagai contoh, Anda memiliki dataset berjumlah 5 juta sampel. Dengan proporsi pembagian 80:20, maka data uji akan berjumlah 1 juta sampel. Tentu ini merupakan jumlah yang terlalu banyak karena kita tidak membutuhkan 1 juta sampel hanya untuk proses pengujian. Dalam kasus proses pengujian ini sebenarnya kita cukup menggunakan 1-2% data atau sebanyak 100.000 hingga 200.000 sampel saja.

Pada modul ini, kita akan menggunakan proporsi pembagian sebesar 90:10 dengan fungsi train\_test\_split dari sklearn.

```
1. from sklearn.model_selection import train_test_split  
2.  
3. X = diamonds.drop(["price"], axis =1)  
4. y = diamonds["price"]  
5. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = 123)
```

Untuk mengecek jumlah sampel pada masing-masing bagian, kita gunakan code berikut.

```
1. print(f'Total # of sample in whole dataset: {len(X)}')  
2. print(f'Total # of sample in train dataset: {len(X_train)}')  
3. print(f'Total # of sample in test dataset: {len(X_test)}')
```

Hasilnya:

```
Total # of sample in whole dataset: 47524  
Total # of sample in train dataset: 42771  
Total # of sample in test dataset: 4753
```

## Standarisasi

Algoritma machine learning memiliki performa lebih baik dan konvergen lebih cepat ketika dimodelkan pada data dengan skala relatif sama atau mendekati distribusi normal. Proses scaling dan standarisasi membantu untuk membuat fitur data menjadi bentuk yang lebih mudah diolah oleh algoritma.

Standardisasi adalah teknik transformasi yang paling umum digunakan dalam tahap persiapan pemodelan. Untuk fitur numerik, kita tidak akan melakukan transformasi dengan one-hot-encoding seperti pada fitur kategori. Kita akan menggunakan teknik StandardScaler dari library Scikitlearn,

StandardScaler melakukan proses standarisasi fitur dengan mengurangkan mean (nilai rata-rata) kemudian membaginya dengan standar deviasi untuk menggeser distribusi. StandardScaler menghasilkan distribusi dengan standar deviasi sama dengan 1 dan mean sama dengan 0. Sekitar 68% dari nilai akan berada di antara -1 dan 1.

Untuk menghindari kebocoran informasi pada data uji, kita hanya akan menerapkan fitur standarisasi pada data latih. Kemudian, pada tahap evaluasi, kita akan melakukan standarisasi pada data uji. Untuk lebih jelasnya, mari kita terapkan StandardScaler pada data.

```
1. from sklearn.preprocessing import StandardScaler
2.
3. numerical_features = ['carat', 'table', 'dimension']
4. scaler = StandardScaler()
5. scaler.fit(X_train[numerical_features])
6. X_train[numerical_features] = scaler.transform(X_train.loc[:, numerical_features])
7. X_train[numerical_features].head()
```

Output:

	carat	table	dimension
536	-0.026226	0.864091	0.143464
21293	1.348407	1.359644	1.353588
45577	-0.511390	-0.622566	-0.372761
37379	-0.834833	-0.622566	-0.905790
38240	-0.861787	-0.622566	-0.813165

Seperti yang telah disebutkan sebelumnya, proses standarisasi mengubah nilai rata-rata (mean) menjadi 0 dan nilai standar deviasi menjadi 1. Untuk mengecek nilai mean dan standar deviasi pada setelah proses standarisasi, jalankan kode ini:

```
1. X_train[numerical_features].describe().round(4)
```

Output:

	carat	table	dimension
count	42771.0000	42771.0000	42771.0000
mean	0.0000	-0.0000	0.0000
std	1.0000	1.0000	1.0000
min	-1.3739	-2.6048	-1.8867
25%	-0.8887	-0.6226	-0.9283
50%	-0.2688	-0.1270	-0.1063
75%	0.8093	0.8641	0.8847
max	3.4777	3.0941	2.6998

Perhatikan tabel di atas, sekarang nilai mean = 0 dan standar deviasi = 1.

Sampai di tahap ini, data kita telah siap untuk dilatih menggunakan model machine learning. Yay! Siap lanjut ke modul berikutnya?

## **Model Development**

Sampailah kita pada tahap pengembangan model setelah semua proses yang kita lewati dari business understanding, data understanding, dan data preparation. Model development adalah tahapan di mana kita menggunakan algoritma machine learning untuk menjawab problem statement dari tahap business understanding.

Pada tahap ini, kita akan mengembangkan model machine learning dengan tiga algoritma. Kemudian, kita akan mengevaluasi performa masing-masing algoritma dan menentukan algoritma mana yang memberikan hasil prediksi terbaik. Ketiga algoritma yang akan kita gunakan, antara lain:

1. K-Nearest Neighbor
2. Random Forest
3. Boosting Algorithm

Apakah Anda sudah familiar menggunakan algoritma-algoritma di atas? Atau ada yang baru mendengar algoritma-algoritma ini? Tidak apa-apa, kita akan bahas sama-sama.

## Model Development dengan K-Nearest Neighbor

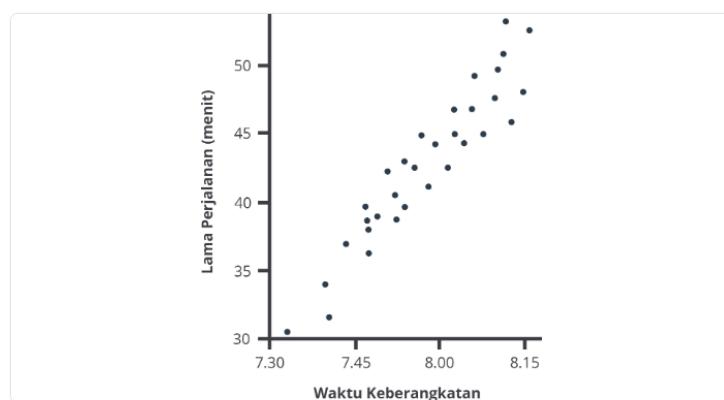
Pertama, kita akan bekerja dengan algoritma K-NN. Sebelum menulis code, Anda perlu memahami apa itu KNN dan bagaimana KNN bekerja.

KNN adalah algoritma yang relatif sederhana dibandingkan dengan algoritma lain. Algoritma KNN menggunakan 'kesamaan fitur' untuk memprediksi nilai dari setiap data yang baru. Dengan kata lain, setiap data baru diberi nilai berdasarkan seberapa mirip titik tersebut dalam set pelatihan.

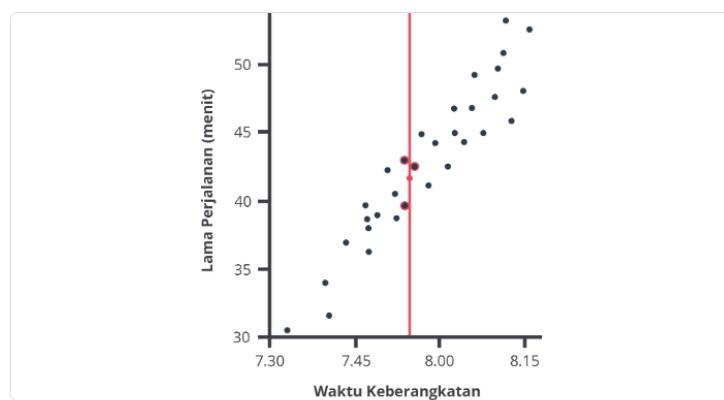
KNN bekerja dengan membandingkan jarak satu sampel ke sampel pelatihan lain dengan memilih sejumlah k tetangga terdekat (dengan k adalah sebuah angka positif). Nah, itulah mengapa algoritma ini dinamakan K-nearest neighbor (sejumlah k tetangga terdekat). KNN bisa digunakan untuk kasus klasifikasi dan regresi. Pada modul ini, kita akan menggunakan untuk kasus regresi.

Bagaimana algoritma KNN bekerja pada kasus regresi? Perhatikan penjelasan dan ilustrasi berikut ya.

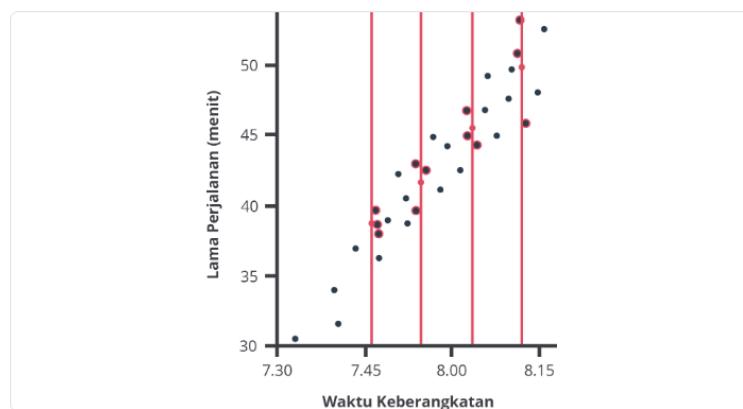
Bayangkan Anda ingin memprediksi berapa lama waktu yang diperlukan untuk perjalanan dari rumah ke kantor. Setiap hari selama beberapa minggu, Anda mencatat waktu berangkat dan lama perjalanan yang dibutuhkan hingga sampai di kantor. Waktu perjalanan Anda tentu dipengaruhi oleh beberapa hal, misalnya arus lalu lintas yang bervariasi di pagi hari. Hal ini menyebabkan waktu perjalanan Anda berubah, tergantung jam berapa Anda berangkat. Contoh grafik waktu keberangkatan dan lama perjalanan dapat dilihat pada gambar berikut.



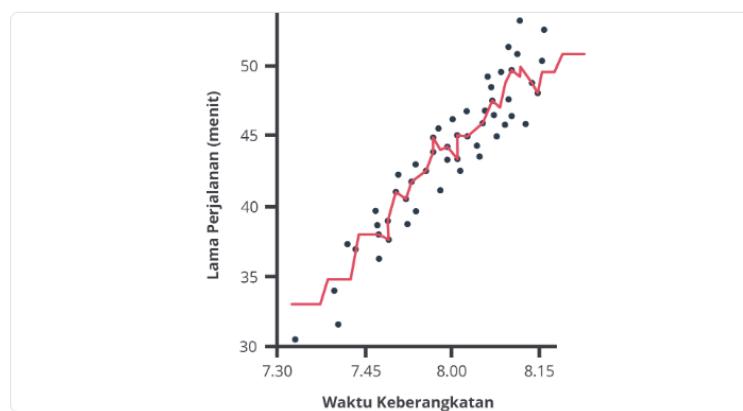
Kemudian, dengan memilih sebuah nilai X pada sumbu Waktu Keberangkatan, kita akan menentukan tiga titik ( $k=3$ ) prediksi paling dekat dengan nilai X yang dipilih. Pada grafik ini ditandai dengan lingkaran merah pada titik hitam. Nilai prediksi terbaik untuk X berdasarkan ketiga titik ini adalah titik kecil berwarna merah. Perhatikanlah titik merah kecil di antara tiga titik hitam dengan lingkaran merah. Titik merah kecil ini diperoleh dari jarak rata-rata ketiga titik hitam dengan lingkaran merah.



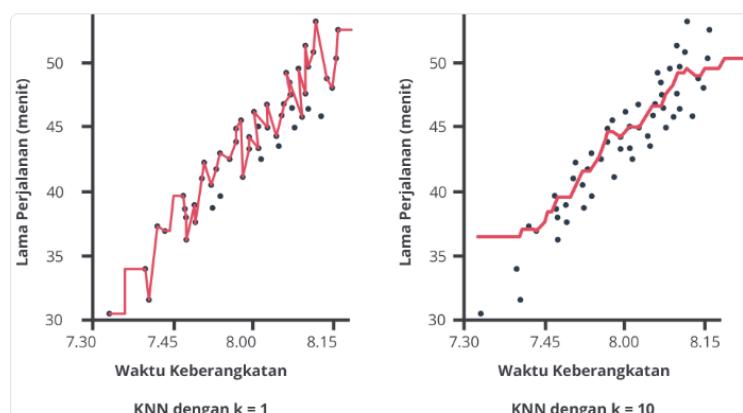
Selanjutnya, bayangkan Anda melakukan ini pada semua kemungkinan nilai masukan sehingga memperoleh nilai prediksi di beberapa tempat.



Tahap terakhir adalah menghubungkan setiap prediksi ini sehingga memberi kita hasil regresi sebagai berikut:



Pada contoh kasus di atas, kita membuat prediksi dengan nilai  $k = 3$ . Memilih nilai  $k$  yang lebih besar dapat membantu menghindari overfit, meskipun kadang bisa menyebabkan kehilangan kemampuan prediksi. Pada prediksi dengan nilai  $k = 1$  (hanya mengecek satu tetangga terdekat), maka hasilnya akan sangat rigid atau kaku. Sedangkan jika kita set nilai  $k$  lebih besar, misalnya pada kasus ini  $k = 10$ , maka hasil prediksi akan lebih smooth (halus).



Pemilihan nilai  $k$  sangat penting dan berpengaruh terhadap performa model. Jika kita memilih  $k$  yang terlalu rendah, maka akan menghasilkan model yang overfit dan hasil prediksinya memiliki varians tinggi. Jika kita memilih  $k$  terlalu tinggi, maka model yang dihasilkan akan underfit dan prediksinya memiliki bias yang tinggi [26]. Namun, kita dapat mencoba beberapa nilai  $k$  yang berbeda, misal: nilai dari 1 hingga 20, kemudian membandingkan mana nilai yang paling sesuai untuk model.

Selanjutnya, untuk menentukan titik mana dalam data yang paling mirip dengan input baru, KNN menggunakan perhitungan ukuran jarak. Metrik ukuran jarak yang digunakan secara default pada library sklearn adalah Minkowski distance [27]. Beberapa metrik ukuran jarak yang juga sering dipakai antara lain: Euclidean distance dan Manhattan distance. Sebagai contoh, jarak Euclidean dihitung sebagai akar kuadrat dari jumlah selisih kuadrat antara titik a dan titik b. Dirumuskan sebagai berikut:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Sedangkan, Minkowski distance merupakan generalisasi dari Euclidean dan Manhattan distance. Untuk menghitungnya, perhatikan rumus berikut:

$$d(x, y) = \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Nah, setelah memahami bagaimana KNN bekerja, yuk kita implementasikan algoritma KNN ke dalam kode.

Seperti yang telah disebutkan pada materi Model Development, kita akan membuat tiga buah model machine learning dengan algoritma berikut:

1. K-Nearest Neighbor (KNN)
2. Random Forest
3. Boosting Algorithm.

Mari siapkan data frame untuk analisis ketiga model tersebut lebih dahulu.

```
1. # Siapkan dataframe untuk analisis model
2. models = pd.DataFrame(index=['train_mse', 'test_mse'],
3.                         columns=['KNN', 'RandomForest', 'Boosting'])
```

Selanjutnya, untuk melatih data dengan KNN, tuliskan code berikut.

```
1. from sklearn.neighbors import KNeighborsRegressor
2. from sklearn.metrics import mean_squared_error
3.
4. knn = KNeighborsRegressor(n_neighbors=10)
5. knn.fit(X_train, y_train)
6.
7. models.loc['train_mse', 'knn'] = mean_squared_error(y_pred = knn.predict(X_train), y_true=y_tr
```

Kita menggunakan  $k = 10$  tetangga dan metric Euclidean untuk mengukur jarak antara titik. Pada tahap ini kita hanya melatih data training dan menyimpan data testing untuk tahap evaluasi yang akan dibahas di Modul Evaluasi Model.

Meskipun algoritma KNN mudah dipahami dan digunakan, ia memiliki kekurangan jika dihadapkan pada jumlah fitur atau dimensi yang besar. Permasalahan ini sering disebut sebagai *curse of dimensionality* (kutukan dimensi). Pada dasarnya, permasalahan ini muncul ketika jumlah sampel meningkat secara eksponensial seiring dengan jumlah dimensi (fitur) pada data. Jadi, jika Anda menggunakan model KNN, pastikan data yang digunakan memiliki fitur yang relatif sedikit, ya!

Selanjutnya, kita akan melatih model kita dengan algoritma yang kedua, yaitu Random Forest. Sudah siap untuk mulai?

## Model Development dengan Random Forest

Bayangkan Anda berada di hutan. Berdiri di tengah hijau rerumputan dan rimbun dedaunan. Kabut menyelimuti, tetes embun membasahi telapak kaki. Sayup terdengar kicau burung, deru angin, serta gemicik suara air terjun di kejauhan. Semburat cahaya muncul dari balik pepohonan, membuyarkan lamunan.



Halo! Sadarlah, Anda tengah berada di depan layar saat ini, belajar mengenai algoritma random forest. Tentu saja ini bukan algoritma hutan yang kemunculannya bersifat acak, seperti si dia yang tiba-tiba melintas dalam benak. Random sekali.

Canda, ya. Jadi, apa itu algoritma random forest?

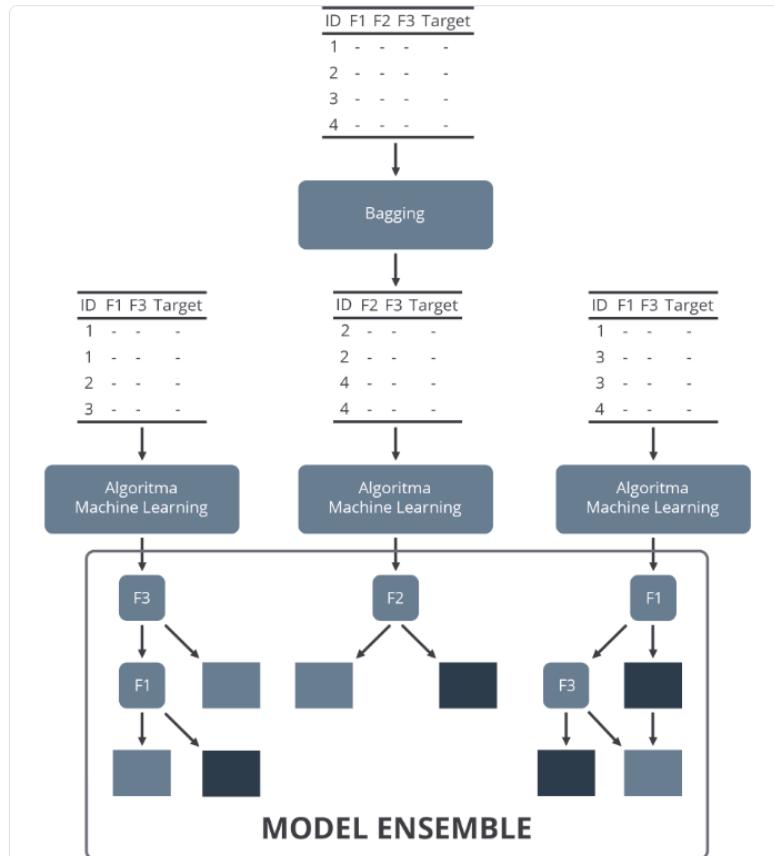
Algoritma random forest adalah salah satu algoritma supervised learning. Ia dapat digunakan untuk menyelesaikan masalah klasifikasi dan regresi. Random forest juga merupakan algoritma yang sering digunakan karena cukup sederhana tetapi memiliki stabilitas yang mumpuni.

Random forest merupakan salah satu model machine learning yang termasuk ke dalam kategori ensemble (group) learning. Apa itu model ensemble? Sederhananya, ia merupakan model prediksi yang terdiri dari beberapa model dan bekerja secara bersama-sama. Ide dibalik model ensemble adalah sekelompok model yang bekerja bersama menyelesaikan masalah. Sehingga, tingkat keberhasilan akan lebih tinggi dibanding model yang bekerja sendirian. Pada model ensemble, setiap model harus membuat prediksi secara independen. Kemudian, prediksi dari setiap model ensemble ini digabungkan untuk membuat prediksi akhir.

Ada dua teknik pendekatan dalam membuat model ensemble, yaitu bagging dan boosting. Jangan bingung dulu dengan istilah ini ya. Kita akan bahas satu per satu.

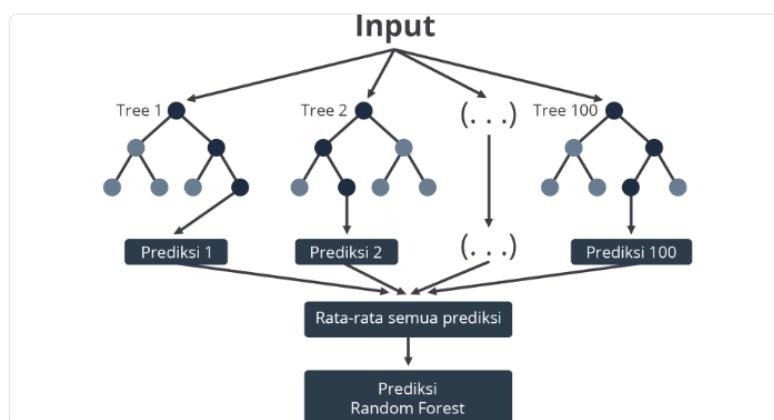
Bagging atau bootstrap aggregating adalah teknik yang melatih model dengan sampel random. Dalam teknik bagging, sejumlah model dilatih dengan teknik *sampling with replacement* (proses sampling dengan penggantian) [19]. Ketika kita melakukan sampling with replacement, sampel dengan nilai yang berbeda bersifat independen. Artinya, nilai suatu sampel tidak mempengaruhi sampel lainnya [28]. Akibatnya, model yang dilatih akan berbeda antara satu dan lainnya.

Berikut adalah ilustrasi pembuatan model ensemble dengan teknik bagging [19].



Algoritma yang cocok untuk teknik bagging ini adalah decision tree. Nah, random forest pada dasarnya adalah versi bagging dari algoritma decision tree. Bayangkan Anda memiliki satu bag (tas) random forest yang berisi beberapa model decision tree. Model decision tree masing-masing memiliki hyperparameter yang berbeda dan dilatih pada beberapa bagian (subset) data yang berbeda juga. Teknik pembagian data pada algoritma decision tree adalah memilih sejumlah fitur dan sejumlah sampel secara acak dari dataset yang terdiri dari n fitur dan m sampel.

Misalnya, ada 100 model decision tree pada bag random forest kita, ini berarti bahwa keputusan (decision) yang dibuat oleh setiap pohon (model) akan sangat bervariasi. Pada kasus klasifikasi, prediksi akhir diambil dari prediksi terbanyak pada seluruh pohon. Sedangkan, pada kasus regresi, prediksi akhir adalah rata-rata prediksi seluruh pohon dalam model ensemble.



Sekarang tentu Anda mengerti, mengapa algoritma ini disebut sebagai random forest. Karena algoritma ini disusun dari banyak algoritma pohon (decision tree) yang pembagian data dan fiturnya dipilih secara acak.

Sekarang tentu Anda mengerti, mengapa algoritma ini disebut sebagai random forest. Karena algoritma ini disusun dari banyak algoritma pohon (decision tree) yang pembagian data dan fiturnya dipilih secara acak.

Sampai di sini, mari kita lanjut menerapkan algoritma ini pada dataset menggunakan library scikit-learn.

```
1. # Impor library yang dibutuhkan
2. from sklearn.ensemble import RandomForestRegressor
3.
4. # buat model prediksi
5. RF = RandomForestRegressor(n_estimators=50, max_depth=16, random_state=55, n_jobs=-1)
6. RF.fit(X_train, y_train)
7.
8. models.loc['train_mse', 'RandomForest'] = mean_squared_error(y_pred=RF.predict(X_train), y_true
```

Perhatikanlah potongan kode di atas. Pertama, Anda mengimpor **RandomForestRegressor** dari library scikit-learn. Anda juga mengimpor **mean\_squared\_error** sebagai metrik untuk mengevaluasi performa model. Selanjutnya, Anda membuat variabel RF dan memanggil **RandomForestRegressor** dengan beberapa nilai parameter. Berikut adalah parameter-parameter yang digunakan:

- **n\_estimator**: jumlah trees (pohon) di forest. Di sini kita set **n\_estimator**=50.
- **max\_depth**: kedalaman atau panjang pohon. Ia merupakan ukuran seberapa banyak pohon dapat membelah (splitting) untuk membagi setiap node ke dalam jumlah pengamatan yang diinginkan.
- **random\_state**: digunakan untuk mengontrol random number generator yang digunakan.
- **n\_jobs**: jumlah job (pekerjaan) yang digunakan secara paralel. Ia merupakan komponen untuk mengontrol thread atau proses yang berjalan secara paralel. **n\_jobs=-1** artinya semua proses berjalan secara paralel.

Setelah model ini dijalankan (run), simpan dulu hasilnya untuk tahap evaluasi nanti. Sekarang, kita akan lanjut pada algoritma ketiga yaitu algoritma boosting di materi berikutnya. Semangat!

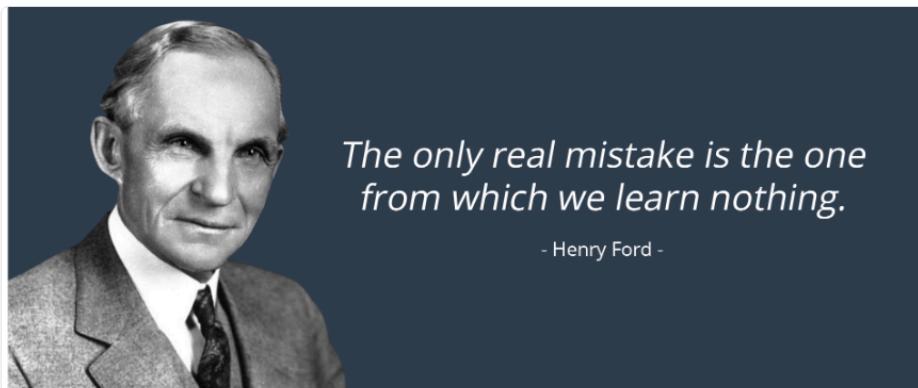
## Model Development dengan Boosting Algorithm

Pada modul sebelumnya, Anda telah belajar mengenai algoritma bagging. Selanjutnya, kita akan membahas mengenai algoritma boosting. Seperti yang dinyatakan pada bahasan sebelumnya, bagging dan boosting merupakan metode ensemble learning. Lalu, apa perbedaan antara bagging dan boosting?

Sebagai model ensemble, keduanya terdiri dari beberapa model yang bekerja secara bersama-sama. Pada teknik bagging, setiap model dilatih secara paralel. Sedangkan, pada teknik boosting, model dilatih secara berurutan atau dalam proses yang iteratif.

Algoritma yang menggunakan teknik boosting bekerja dengan membangun model dari data latih. Kemudian ia membuat model kedua yang bertugas memperbaiki kesalahan dari model pertama. Model ditambahkan sampai data latih terprediksi dengan baik atau telah mencapai jumlah maksimum model untuk ditambahkan.

Model saja belajar dari kesalahan untuk memperbaiki diri, bagaimana dengan kita?



Seperti namanya, boosting, algoritma ini bertujuan untuk meningkatkan performa atau akurasi prediksi. Caranya adalah dengan menggabungkan beberapa model sederhana dan dianggap lemah (weak learners) sehingga membentuk suatu model yang kuat (strong ensemble learner). Algoritma boosting muncul dari gagasan mengenai apakah algoritma yang sederhana seperti linear regression dan decision tree dapat dimodifikasi untuk dapat meningkatkan performa.

Algoritma boosting telah ada sejak puluhan tahun lalu. Namun, baru beberapa tahun ini ramai dibicarakan. Mengapa algoritma ini begitu populer akhir-akhir ini?

Salah satu alasannya adalah peningkatan algoritma boosting dalam kompetisi machine learning atau data science. Algoritma ini sangat powerful dalam meningkatkan akurasi prediksi. Algoritma boosting sering mengungguli model yang lebih sederhana seperti logistic regression dan random forest. Beberapa pemenang kompetisi di platform Kaggle menyatakan bahwa mereka menggunakan algoritma boosting atau kombinasi beberapa algoritma boosting dalam modelnya. Meskipun demikian, hal ini tetap bergantung pada kasus per kasus, ruang lingkup masalah, dan dataset yang digunakan.

Dilihat dari caranya memperbaiki kesalahan pada model sebelumnya, algoritma boosting terdiri dari dua metode:

- Adaptive boosting
- Gradient boosting

Pada modul ini, kita akan menggunakan metode adaptive boosting. Salah satu metode adaptive boosting yang terkenal adalah AdaBoost, dikenalkan oleh Freund and Schapire (1995) [19].

Pertanyaannya adalah, bagaimana AdaBoost bekerja?

Awalnya, semua kasus dalam data latih memiliki weight atau bobot yang sama. Pada setiap tahapan, model akan memeriksa apakah observasi yang dilakukan sudah benar? Bobot yang lebih tinggi kemudian diberikan pada model yang salah sehingga mereka akan dimasukkan ke dalam tahapan selanjutnya. Proses iteratif ini berlanjut sampai model mencapai akurasi yang diinginkan.

Untuk menerapkan model pada dataset, implementasikan code berikut.

```
1. from sklearn.ensemble import AdaBoostRegressor  
2.  
3. boosting = AdaBoostRegressor(learning_rate=0.05, random_state=55)  
4. boosting.fit(X_train, y_train)  
5. models.loc['train_mse', 'Boosting'] = mean_squared_error(y_pred=boosting.predict(X_train), y_tr
```

Berikut merupakan [parameter-parameter](#) yang digunakan pada potongan kode di atas.

- learning\_rate: bobot yang diterapkan pada setiap regressor di masing-masing proses iterasi boosting.
- random\_state: digunakan untuk mengontrol random number generator yang digunakan.

Kita simpan hasilnya untuk nanti. Sekarang, mari kita lanjutkan pada tahap evaluasi model.

## Evaluasi Model

Mengevaluasi model regresi sebenarnya relatif sederhana. Secara umum, hampir semua metrik adalah sama. Jika prediksi mendekati nilai sebenarnya, performanya baik. Sedangkan jika tidak, performanya buruk. Secara teknis, selisih antara nilai sebenarnya dan nilai prediksi disebut eror. Maka, semua metrik mengukur seberapa kecil nilai eror tersebut.

Metrik yang akan kita gunakan pada prediksi ini adalah MSE atau Mean Squared Error yang menghitung jumlah selisih kuadrat rata-rata nilai sebenarnya dengan nilai prediksi. MSE didefinisikan dalam persamaan berikut

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - y_{pred_i})^2$$

Keterangan:

$N$  = jumlah dataset

$y_i$  = nilai sebenarnya

$y_{pred}$  = nilai prediksi

Namun, sebelum menghitung nilai MSE dalam model, kita perlu melakukan proses scaling fitur numerik pada data uji. Sebelumnya, kita baru melakukan proses scaling pada data latih untuk menghindari kebocoran data. Sekarang, setelah model selesai dilatih dengan 3 algoritma, yaitu KNN, Random Forest, dan Adaboost, kita perlu melakukan proses scaling terhadap data uji. Hal ini harus dilakukan agar skala antara data latih dan data uji sama dan kita bisa melakukan evaluasi.

Untuk proses scaling, implementasikan kode berikut:

```
1. # Lakukan scaling terhadap fitur numerik pada X_test sehingga memiliki rata-rata=0 dan varians=1
2. X_test.loc[:, numerical_features] = scaler.transform(X_test[numerical_features])
```

Selanjutnya, mari kita evaluasi ketiga model kita dengan metrik MSE yang telah dijelaskan di atas. Jalankan kode berikut.

```
1. # Buat variabel mse yang isinya adalah dataframe nilai mse data train dan test pada masing-masing algoritma
2. mse = pd.DataFrame(columns=['train', 'test'], index=['KNN','RF','Boosting'])
3.
4. # Buat dictionary untuk setiap algoritma yang digunakan
5. model_dict = {'KNN': knn, 'RF': RF, 'Boosting': boosting}
6.
7. # Hitung Mean Squared Error masing-masing algoritma pada data train dan test
8. for name, model in model_dict.items():
9.     mse.loc[name, 'train'] = mean_squared_error(y_true=y_train, y_pred=model.predict(X_train))
10.    mse.loc[name, 'test'] = mean_squared_error(y_true=y_test, y_pred=model.predict(X_test))/1e3
11.
12. # Panggil mse
13. mse
14.
```

Perhatikanlah potongan kode di atas. Saat menghitung nilai Mean Squared Error pada data train dan test, kita membaginya dengan nilai 1e3. Hal ini bertujuan agar nilai mse berada dalam skala yang tidak terlalu besar.

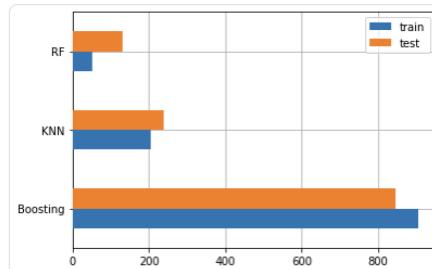
Hasil evaluasi pada data latih dan data test adalah sebagai berikut.

	train	test
KNN	203.762502	239.539894
RF	52.287366	130.788418
Boosting	904.838013	846.212966

Untuk memudahkan, mari kita plot metrik tersebut dengan bar chart. Implementasikan kode di bawah ini:

```
1. fig, ax = plt.subplots()
2. mse.sort_values(by='test', ascending=False).plot(kind='barh', ax=ax, zorder=3)
3. ax.grid(zorder=0)
```

Hasilnya sebagai berikut.



Dari gambar di atas, terlihat bahwa, model Random Forest (RF) memberikan nilai eror yang paling kecil. Sedangkan model dengan algoritma Boosting memiliki eror yang paling besar (berdasarkan grafik, angkanya di atas 800). Sehingga model RF yang akan kita pilih sebagai model terbaik untuk melakukan prediksi harga diamonds.

Untuk mengujinya, mari kita buat prediksi menggunakan beberapa harga dari data test.

```
1. prediksi = X_test.iloc[:1].copy()
2. pred_dict = {'y_true':y_test[:1]}
3. for name, model in model_dict.items():
4.     pred_dict['prediksi_'+name] = model.predict(prediksi).round(1)
5.
6. pd.DataFrame(pred_dict)
```

Hasilnya adalah sebagai berikut:

y_true	prediksi_KNN	prediksi_RF	prediksi_Boosting
29703	886	923.2	884.2

Terlihat bahwa prediksi dengan Random Forest (RF) memberikan hasil yang paling mendekati. Anda juga bisa menguji hasil prediksi pada data lain dengan cara mengubah indeks pada `X_test`. Jika Anda merasa belum yakin dengan model ini, lakukanlah pengaturan parameter pada algoritma yang digunakan. Sebagai contoh, pada algoritma Random Forest, Anda dapat mengubah berbagai parameter seperti `n_estimators` (jumlah pohon/trees dalam forest), atau parameter lain. Anda dapat merujuk pada dokumentasi [Random Forest](#) pada library `sklearn` untuk pengaturan parameter lainnya.

Untuk melakukan peningkatan performa, lakukanlah hal yang sama (pengaturan parameter) pada semua algoritma yang digunakan. Selain itu, Anda juga dapat melakukan optimasi parameter dengan menerapkan teknik Grid Search.

## Rangkuman Studi Kasus Pertama: Predictive Analytics

Anda berada di akhir dari modul Studi Kasus Pertama: Predictive Analytics. Kita sudah belajar banyak hal baru dalam modul ini. Mari kita uraikan apa saja materi yang telah dibahas di modul ini.

Predictive analytics adalah bidang terapan yang menggunakan berbagai metode kuantitatif untuk membuat prediksi dengan memanfaatkan data. Ia merupakan seni membangun serta menggunakan model prediksi berdasarkan pola data historis. Data diubah dan diekstrak menjadi informasi berharga untuk menghasilkan keputusan analytics. Saat ini, keputusan analytics merupakan senjata strategis di bidang bisnis.

Keberhasilan proyek predictive analytics tidak hanya ditentukan oleh pemilihan algoritma machine learning saja, melainkan juga penerapan metodologi standar dalam mengelola seluruh tahapan atau siklus proyek. Cross-Industry Standard Process for Data Mining atau disingkat menjadi CRISP-DM merupakan salah satu metode standar proses analitik yang paling umum digunakan.

Berikut merupakan fase-fase utama proses analitik dalam metode CRISP-DM:

### 1. Business understanding.

Fase atau tahapan ini mencakup proses klarifikasi masalah, menentukan tujuan atau objective, dan mengidentifikasi sumber daya yang dimiliki.

### 2. Data understanding.

Ini merupakan tahapan untuk memahami informasi dalam data dan menentukan kualitasnya.

### 3. Data preparation.

Pada tahapan ini, Anda melakukan proses transformasi pada data sehingga menjadi bentuk yang cocok untuk proses pemodelan.

### 4. Modeling.

Ini merupakan tahapan saat Anda menggunakan algoritma machine learning untuk membuat model prediksi.

### 5. Evaluation.

Tahapan ini bertujuan untuk memastikan bahwa model akan mampu membuat prediksi yang akurat dan tidak mengalami overfitting atau underfitting.

### 6. Deployment.

Pada tahap ini, model machine learning yang telah Anda buat akan diintegrasikan ke dalam sistem organisasi.

Pada modul ini, Anda telah belajar menerapkan metode CRISP-DM dengan studi kasus prediksi harga diamonds. Pada studi kasus ini, Anda menggunakan tiga jenis algoritma, yaitu:

- **K-Nearest Neighbor.**

KNN bekerja dengan membandingkan jarak satu sampel ke sampel pelatihan lain dengan memilih sejumlah k-tetangga terdekat. Algoritma KNN menggunakan 'kesamaan fitur' untuk memprediksi nilai dari setiap data yang baru. Dengan kata lain, setiap data baru diberi nilai berdasarkan seberapa mirip titik tersebut dalam set pelatihan.

- **Random Forest.**

Algoritma random forest adalah salah satu algoritma supervised learning yang dapat digunakan untuk menyelesaikan masalah klasifikasi dan regresi. Ia termasuk ke dalam kelompok model ensemble (group). Algoritma ini disusun dari banyak algoritma pohon (decision tree) yang pembagian data dan fiturnya dipilih secara acak.

- **Boosting Algorithm.**

Algoritma boosting bekerja dengan membangun model dari data latih. Kemudian ia membuat model kedua yang bertugas memperbaiki kesalahan dari model pertama. Model ditambahkan sampai data latih terprediksi dengan baik atau telah mencapai jumlah maksimum model untuk ditambahkan.

Dengan uraian tersebut, diharapkan Anda dapat memahami apa itu model predictive analytics dan langkah-langkah dalam membuat modelnya. Jika ada materi yang masih belum dipahami, Anda bisa mengulas kembali materi yang diberikan di modul ini atau menanyakan di forum diskusi.

## **Daftar Referensi**

Jika tertarik untuk mengeksplorasi lebih lanjut mengenai materi Predictive Analytics Anda dapat mempelajarinya melalui daftar referensi berikut:

- [19] Kelleher, John D, et al. "Machine Learning for Predictive Data Analytics". MIT Press. 2020. Tersedia: [tautan informasi buku](#).
- [20] Delen, Dursun. "Predictive Analytic: Data Mining, Machine Learning and Data Science for Practitioners". Pearson FT. Press. 2020. Tersedia: [O'Reilly media](#).
- [21] IBM Cloud. "Exploratory Data Analysis". Tersedia: [tautan](#). Diakses pada 2 Juni 2021.
- [22] Kang, Hyun. "The Prevention and Handling the Missing Data". Tersedia: [tautan](#). Diakses pada Juni 2021.
- [23] Kuhn, Max dan Johnson Kjell. "Applied Predictive Modeling". Springer. 2013.
- [24] Seltman, Howard J. "Experimental Design and Analysis". 2018. Tersedia: [tautan](#). Diakses pada Juni 2021.
- [25] Fuentes, Alvaro. "Hands-on Predictive Analytics with Python". Packt Publishing. 2018. Tersedia: [O'Reilly Media](#).
- [26] Rhys, Hefin. "Machine Learning with R, the Tidyverse, and MLR". Manning Publications. 2020. Tersedia: [O'Reilly Media](#).
- [27] Scikit-learn Documentation. Tersedia: [tautan](#). Diakses pada: Juni 2021.
- [28] Parker, Mary. "Sampling with Replacement and Sampling without Replacement". Tersedia: [tautan](#). Diakses pada Juni 2021.