

Glosarium

Berikut adalah glosarium dengan istilah umum yang digunakan pada kelas ini. Anda dapat membaca sekilas materi berikut untuk mengenali istilah-istilah umum yang ada di modul kelas ini. Selain itu, Anda juga dapat mengunjungi kembali halaman ini setiap kali menemukan istilah yang belum dimengerti. Carilah istilah tersebut pada halaman glosarium ini untuk mengidentifikasi makna atau definisinya. Jika masih terdapat kosakata yang tidak Anda pahami dan belum masuk di daftar ini, Anda dapat memberikan saran melalui fitur Laporan Materi.

A

Action

Dalam reinforcement learning, action adalah setiap keputusan yang diambil oleh agent.

Activation Function

Fungsi yang menerima jumlah bobot semua masukan dari layer sebelumnya, kemudian menghasilkan dan meneruskan nilai keluaran ke layer berikutnya. Contoh: ReLU atau sigmoid.

Akurasi

Jumlah data yang diprediksi dengan benar oleh machine learning dibagi jumlah seluruh data poin.

Algoritma

Dalam ilmu komputer, algoritma adalah sekumpulan aturan atau instruksi yang didesain untuk melakukan tugas dan menyelesaikan permasalahan.

Anomaly Detection

Proses untuk menemukan anomali pada dataset.

Artificial Intelligence

Program atau model yang dapat menyelesaikan tugas-tugas canggih. Machine learning adalah sub-bidang dari artificial intelligence.

B

Bobot

Koefisien untuk fitur dalam model linier atau parameter yang dipelajari oleh sebuah perceptron untuk menunjukkan kekuatan node tertentu dalam jaringan syaraf tiruan.

C

Clustering

Pengelompokan data yang memiliki kesamaan ke dalam grup tertentu.

CSV

Comma-Separated Value, format file teks dalam bentuk tabular dengan pemisah berupa koma

D

Data Kategorik

Fitur-fitur yang memiliki sekumpulan nilai diskrit dan bisa dibagi ke dalam grup. Sering disebut juga data diskrit.

Data Numerik

Fitur-fitur yang direpresentasikan sebagai bilangan bulat atau bilangan riil. Sering disebut juga fitur berkelanjutan.

Data Time Series

Sekumpulan data atau observasi pada interval waktu tertentu.

Dataset

Sekumpulan data atau contoh-contoh yang terdiri dari satu atau lebih fitur.

Deep Learning

Cabang machine learning dengan algoritma jaringan syaraf tiruan yang dapat belajar dan beradaptasi terhadap sejumlah besar data.

Dimension Reduction

Mengurangi jumlah dimensi yang digunakan untuk merepresentasikan fitur tertentu.

E

Elbow

Metode untuk menentukan jumlah cluster pada dataset.

Epoch

Satu proses pelatihan penuh atas seluruh dataset sehingga setiap contoh data telah melewati model sebanyak satu kali.

Environment

Dalam reinforcement learning, environment adalah sarana untuk berinteraksi, yang dapat menerima action dan memberikan respon berupa hasil maupun data berupa satu set observasi baru.

F

Feedback

Dalam machine learning, situasi di mana prediksi model mempengaruhi data pelatihan.

Fitur

Input variabel yang digunakan untuk membuat prediksi.

G

Good Fit

Kondisi saat model machine learning melakukan generalisasi data dengan baik.

Google Colaboratory

Notebooks yang memungkinkan kita menulis dan mengeksekusi code pada browser.

H

Hidden Layer

Lapisan sintetis dalam jaringan saraf antara lapisan masukan (fitur) dan lapisan keluaran (prediksi). Hidden layer biasanya berisi fungsi aktivasi untuk pelatihan.

Hyperparameter

Variabel yang digunakan untuk mengontrol proses pelatihan model. Contohnya: epoch.

I

Input Layer

Layer pertama yang menerima data masukan dalam jaringan saraf tiruan

K

Keras

API Python machine learning yang populer. Keras bisa dijalankan pada beberapa kerangka kerja deep learning, termasuk TensorFlow yang tersedia sebagai `tf.keras`.

Klasifikasi

Tipe model machine learning untuk membedakan antara dua atau lebih kelas. Misalnya klasifikasi apakah suatu email merupakan email spam atau bukan.

L

Label

Dalam supervised learning, label adalah “jawaban” atau “hasil” dari sebuah contoh.

Layer

Sekumpulan neuron dalam jaringan saraf yang memproses fitur-fitur masukan atau keluaran dari neuron tersebut.

Linear Regression

Menggunakan keluaran y' dari model linier sebagai prediksi aktual dalam model regresi. Tujuan dari permasalahan regresi adalah untuk membuat prediksi yang bernilai benar.

Logistic Regression

Model klasifikasi yang menggunakan fungsi sigmoid untuk mengubah prediksi y' pada model linier menjadi nilai antara 0 dan 1.

M

Machine Learning

Bidang studi yang memberi komputer kemampuan untuk belajar tanpa diprogram secara eksplisit.

Matplotlib

Open-source library pada Python untuk membuat plot 2D. Matplotlib membantu Anda memvisualisasikan berbagai aspek dalam machine learning.

Missing Value

Tidak tersedia nilai data pada variabel tertentu dalam sebuah observasi.

Model

Representasi dari apa yang telah dipelajari oleh sistem machine learning dari data pelatihan. Model mendefinisikan relasi antara fitur dan label.

N

Neural Network

Sebuah model yang mengambil inspirasi dari otak, terdiri dari beberapa layers yang memiliki neuron-neuron yang saling terhubung.

Normalization

Proses mengonversi rentang nilai aktual menjadi rentang nilai standar, biasanya dari -1 hingga +1, atau 0 hingga 1.

Numpy

Library matematika open-source yang menyediakan operasi array efisien dengan Python. Pandas dibangun di atas NumPy.

O

One-hot-encoding

Mengubah data kategorik dengan membuat kolom baru untuk setiap kategori.

Outlier

Sebuah nilai yang jauh berbeda dari kumpulan nilai lainnya dan dapat mengacaukan hasil dari sebuah analisis statistik.

Output layer

Lapisan “terakhir” dari jaringan saraf tiruan. Lapisan yang berisi jawaban.

Overfitting

Kondisi model sangat cocok dengan data pelatihan, tetapi model gagal membuat prediksi yang benar pada data baru.

P

Pandas

API analisis data yang berorientasi pada kolom. Banyak framework machine learning termasuk TensorFlow mendukung struktur data panda sebagai input.

Parameter

Variabel model yang dilatih oleh sistem machine learning. Sebagai contoh: weight/bobot adalah parameter yang nilainya dipelajari secara bertahap oleh sistem machine learning.

Propagasi balik

Algoritma untuk melakukan penurunan gradien pada jaringan saraf tiruan. Pertama, nilai keluaran dari setiap node dihitung dalam sebuah forward pass. Kemudian, turunan

parsial dari eror/kesalahan yang terkait dengan setiap parameter dihitung dalam hitungan mundur melalui grafik.

R

Reinforcement Learning

Algoritma yang belajar menggunakan sistem reward dan pinalti. Algoritma Reinforcement Learning belajar agar terus mendapatkan reward dan menghindari penalti.

Reward

Dalam reinforcement learning, reward adalah hasil numerik dari mengambil tindakan dalam suatu keadaan seperti yang didefinisikan oleh environment. Reward diberikan saat agent berhasil menyelesaikan tantangan.

S

Scikit learn

Platform machine learning open-source yang populer. Lihat www.scikit-learn.org.

SVM

Algoritma training yang bertujuan untuk memaksimalkan margin antara pola pelatihan dan batas keputusan (decision boundary).

T

Tensorflow

Platform machine learning berskala besar dan terdistribusi. Istilah ini juga merujuk pada layer API dasar yang mendukung komputasi umum pada grafik aliran data.

Test set

Bagian dari dataset yang digunakan untuk menguji sebuah model setelah model melalui pemeriksaan awal oleh validation set.

Training set

Bagian dari dataset yang digunakan untuk melatih model.

U

Underfitting

Model machine learning memiliki kemampuan prediksi yang buruk karena model tersebut belum menangkap kompleksitas data pelatihan.

Unsupervised learning

Model machine learning yang belajar dan menemukan pola dalam sekumpulan data tanpa label.

V

Validation set

Bagian dari dataset (berbeda dari training set) yang digunakan sebagai validasi.

Daftar Referensi

- [1] L. Moroney dalam "Course: Introduction to Tensorflow for Artificial Intelligence". Diakses pada: 20 November 2020. [Online Video]. Tersedia di : [tautan](#)
- [2] S. Campbell, et al., "Deep Learning vs. Traditional Computer Vision". Tersedia di : [tautan](#).
- [3] Andreas C. Muller and Sarah Guido, "Introduction to Machine Learning with Python". O'Reilly Media, 2016, Chapter 1.
- [4] Aurelien Geron, "Hands-On Machine Learning with Scikit-Learn & Tensorflow", O'Reilly, 2017.
- [5] Phil Winder, "Reinforcement Learning", O'Reilly Media, Inc, 2020, chapter 1.
- [6] S. K. Srivatsava, et al., "Statistical Data Classification Using Instance Based Learning Algorithm". 2019. International Journal of Scientific & Technology Research Volume 8. ISSN 2277-8616. Tersedia di : [tautan](#).
- [7] Maxima Lapan, "Deep Reinforcement Learning Hands-On 2nd Edition", Packt Publishing, 2020, chapter 2.
- [8] A. Burkov. "The Hundred Page Machine Learning Book". 2019.
- [9] Bayesian and Probabilistic PCA: Examples. An Applied Statistic. Course by Susan Holmes. Tersedia di: [tautan](#)
- [10] V.N.Vapnik, I.M, Guyon, and B.E. Boser., "A Training Algorithm for Optimal margin Classifier", dalam COLT '92: Proceedings of the fifth annual workshop on Computational learning theory. Tersedia di : [tautan](#)
- [11] M. Hachimi, et al., "Multi-stage Jamming Attacks Detection using Deep Learning Combined with Kernelized Support Vector Machine in 5G Cloud Radio Access Networks", dalam IEEE ISNCC'20. Tersedia di : [tautan](#)
- [12] Udiprod. "SVM with Polynomial Kernel Visualization". Feb 5, 2007. Diakses pada 24 Nov 2020. [Online Video]. Tersedia : [tautan](#).
- [13] H. Marius. "Multiclass Classification with Support Vector Machine (SVM), Dual Problem, and Kernel Function". Towards Data Science. [tautan](#) (Diakses pada 22 Nov 2020)

- [14] M. Awad dan R. Khanna, "Efficient Learning Machines", Barkley, CA, USA: Apress, 2015, hlm. 68.
- [15] Jui-Yang Hsia & Chih-Jen Lin, "Parameter Selection for Linear Support Vector Regression". Tersedia di : [link](#).
- [16] K. P. Murphy, "Machine Learning: a Probabilistic Perspective. 2012, page 22
- [17] J. Ding, et al., "Model Selection Technique-An Overview"
- [18] [Brain Basic: The Life and Death of a Neuron](#).
- [19] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". 1958. Psychological Review, Vol 65. No 6, Tersedia di : [tautan](#).
- [20] J. Yang and G. Yang, "Modified Convolutional Neural Network Based on Dropout and the Stochastic Gradient Descent". 2018. MDPI. Tersedia di : [tautan](#)
- [21] David Rumelhart et al. "Learning Internal Representations by Error Propagation", 1985, Defense Technical Information Center technical report. Tersedia di : [tautan](#).
- [22] Y. LeCun, et. al, "Gradient Based Learning Applied to Document Recognition". 1998. Proc of the IEEE. Tersedia di : [tautan](#)
- [23] A. Krizhevsky, et al., "Image Classification with Deep Convolutional Neural Networks", 2012, NIPS. Tersedia di : [tautan](#)
- [24] S. Gass and A. Assad. "Profiles in Operations Research: Pioneers and Innovators". 2011. New York. Springer: 363-386.
- [25] Alberto Boschetti, Luca Massaron. "Python Data Science Essentials Second Edition". Packt Publishing. 2016, 87-88.
- [26] S. Raschka and V. Mirjalili. "Python Machine Learning 3rd Edition". Packt Publishing. 2019. Chapter 4.
- [27] Google Developers, "Introduction to Machine Learning". Tersedia di : [tautan](#).
- [28] <https://doi.org/10.1080/20964471.2019.1572452>.

BAB 1 – PENGENALAN DATA

1.1 Pengenalan Machine Learning

Dalam satu dekade terakhir machine learning menjelma sebagai bidang yang berkembang sangat pesat dan terus dikembangkan para ilmuwan seluruh dunia. Inilah inti dari berbagai macam produk berteknologi tinggi terkini. Perannya sangat signifikan dalam disrupti industri 4.0 yang sarat dengan transformasi digital.

Tahukah Anda, apa itu machine learning?

“A field of study that gives computers the ability to learn without being explicitly programmed.”

--Arthur Samuel, 1959.

Istilah machine learning pertama kali dipopulerkan oleh Arthur Samuel, seorang ilmuwan komputer yang memelopori kecerdasan buatan pada tahun 1959. Menurut Arthur Samuel, machine learning adalah suatu cabang ilmu yang memberi komputer kemampuan untuk belajar tanpa diprogram secara eksplisit. Apa maksudnya?

Mari kita mundur sejenak ke masa sebelum machine learning ditemukan. Seperti dikemukakan oleh Moroney [1], prinsip atau paradigma pemrograman sejak permulaan era komputasi direpresentasikan dalam diagram berikut.



Aturan dan data adalah masukan atau input bagi sistem. Secara eksplisit, **aturan** diekspresikan dalam bahasa pemrograman. Tambahan masukan berupa **data** kemudian akan menghasilkan **solusi** sebagai keluaran. Paradigma pemrograman seperti pada diagram di atas sering disebut sebagai pemrograman tradisional.

Pemrograman tradisional memiliki keterbatasan. Sifatnya rigid dengan sekumpulan aturan “if” dan “else” untuk memproses data atau menyesuaikan dengan masukan.

Sebagai contoh, kita ingin membuat sebuah program untuk mendeteksi aktivitas fisik. Kita bisa menggunakan parameter “kecepatan” sebagai data untuk membedakan aktivitas satu dan lainnya.

Misal untuk aktivitas berjalan, kita membuat algoritma program dengan bahasa python sebagai berikut.

```
1. if kecepatan<4:  
2.     aktivitas=BERJALAN
```

Untuk aktivitas berlari, kecepatannya tentu menjadi lebih besar, misalnya 12 km/jam sehingga algoritmanya menjadi seperti ini.

```
1. if kecepatan<4:  
2.     aktivitas=BERJALAN  
3. else:  
4.     aktivitas=BERLARI
```

Jika kemudian kita ingin mendeteksi aktivitas bersepeda, algoritma program kita menjadi seperti ini:

```
1. if kecepatan<4:  
2.     aktivitas=BERJALAN  
3. elif kecepatan<12:  
4.     aktivitas=BERLARI  
5. else  
6.     aktivitas=BERSEPEDA
```

Cukup mudah, ya? Kita hanya perlu mendeteksi kecepatan seseorang untuk menentukan aktivitas yang sedang dilakukannya. Tapi bagaimana jika kita diminta untuk mendeteksi aktivitas lain seperti “bermain basket”?

Kita akan menemui masalah. Orang yang sedang bermain basket akan melakukan aktivitas berjalan, kadang berlari, sekejap berhenti, dan seterusnya. Lantas, bagaimana

cara program mendeteksi aktivitas tersebut? Hal ini membuat kita menyadari bahwa pemrograman tradisional memiliki keterbatasan dalam menyelesaikan masalah.

Sampai sini jelas, ya? Agar lebih memahami, kita akan berikan satu contoh lain. Misal kita ingin mengetahui bagaimana pemrograman tradisional bekerja untuk mengenali gambar.

Pendekatan tradisional menggunakan teknik *feature extraction* untuk mendeteksi objek. Teknik ini merepresentasikan gambar dengan mengekstrak beberapa fitur pada gambar. Fitur adalah bagian kecil yang menarik, deskriptif, atau informatif. Ia bisa berupa sudut, tepi, skema warna, tekstur gambar, dan lain-lain. Beberapa contoh algoritma feature extraction yang sering digunakan adalah: [Harris Corner Detection](#), Scale-Invariant Feature Transform ([SIFT](#)), Speeded-Up Robust Features ([SURF](#)), Features from Accelerated Segment Test ([FAST](#)), dan Binary Robust Independent Elementary Features ([BRIEF](#)).

S. Campbell dalam makalahnya yang berjudul *Deep Learning vs, Traditional Computer Vision* [2] menyebutkan bahwa fitur-fitur diekstrak dari gambar untuk membentuk definisi dari setiap objek kelas. Dalam tahap penerapan, definisi ini dicari di berbagai tempat pada gambar. Jika sejumlah fitur ditemukan pada gambar lain, gambar diklasifikasikan sebagai gambar yang mengandung objek tertentu.

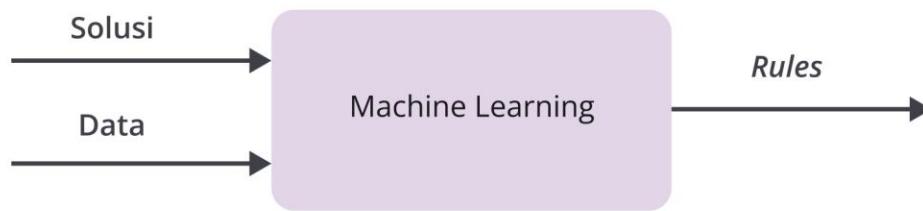
Kesulitan yang muncul dari pendekatan tradisional ini adalah kita perlu memilih mana fitur yang penting di setiap gambar. Seiring dengan meningkatnya jumlah kelas yang akan diklasifikasikan, proses ekstraksi fitur menjadi semakin rumit. Proses ini sangat bergantung pada keahlian manusia untuk menilai fitur mana yang bisa mendeskripsikan kelas yang berbeda. Prosesnya cukup rumit, ya? Tapi tentu ada cara yang lebih baik.

Perkenalkan: paradigma baru pemrograman dengan machine learning!

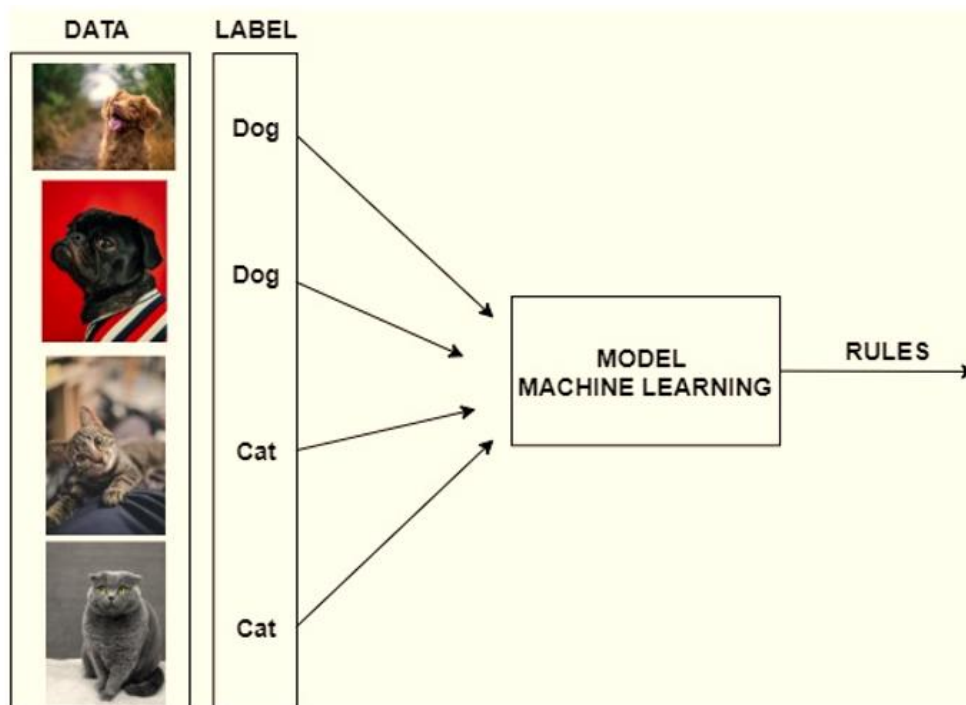
Perhatikan diagram di bawah ini. Paradigma pemrograman pada machine learning itu berkebalikan dengan pemrograman tradisional.

Pada pemrograman tradisional kita merepresentasikan masalah menjadi aturan dalam bahasa pemrograman. Kini ketika hal itu tidak lagi memungkinkan, kita perlu mengubah alur berpikir kita dengan cara yang berbeda. Paradigma baru pemrograman dengan machine learning adalah kita memiliki banyak sekali **data** dan label bagi data tersebut. Kita juga telah mengetahui keterkaitan antara data dengan label sebagai suatu **solusi**.

Ilustrasinya seperti ini, ya.

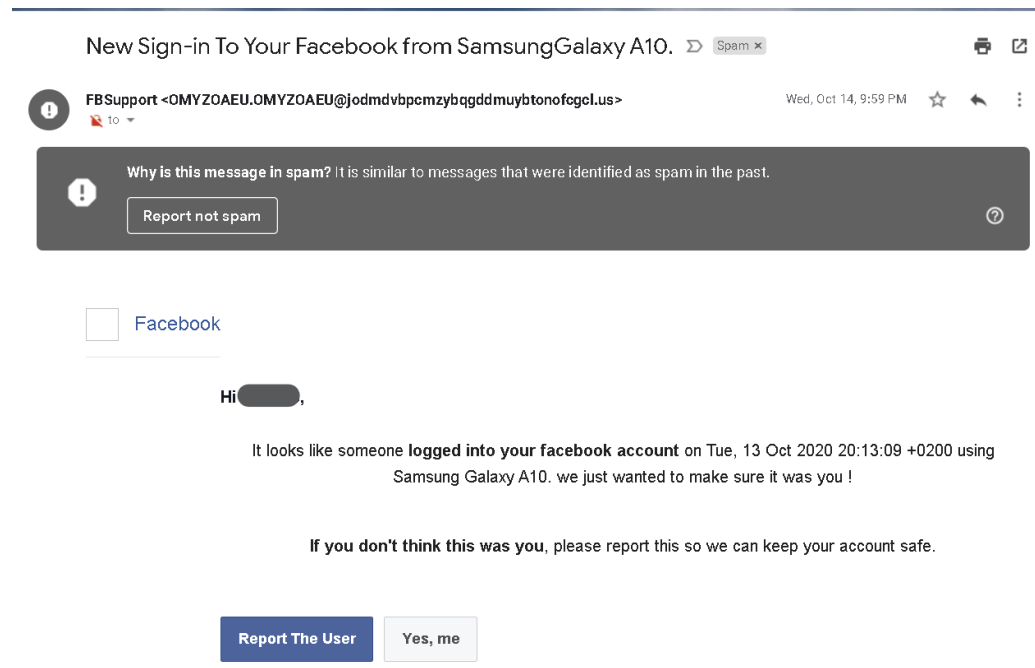


Algoritma machine learning mencari pola tertentu dari setiap kumpulan data yang menentukan kekhasan masing-masing untuk kemudian menyimpulkan sebuah **aturan**. Selanjutnya, aturan ini dapat digunakan untuk melakukan identifikasi dan prediksi bagi data baru yang relevan dengan model yang kita miliki. Sangat powerful dan menarik ya?



Ilustrasi-ilustrasi di atas menjelaskan proses belajar pada pemrograman dengan machine learning. Lantas apa contoh sederhana dari machine learning? Filter spam pada layanan email, misalnya. Saat kita menandai satu email sebagai spam, maka program akan mempelajari anatomi email tersebut untuk mengantisipasi email-email masuk berikutnya itu spam atau bukan. Jika mirip, sebuah email baru akan masuk kategori spam, demikian juga sebaliknya.

Berikut adalah contoh email yang telah ditandai sebagai spam oleh algoritma ML dan masuk ke dalam kotak spam.



Anda tentu pernah mendapati email dalam kotak spam. Uniknya, filter email spam ini bukanlah hal yang baru, melainkan telah ada sejak tahun 1990-an saat internet tengah *booming*. Alhasil, hidup jutaan orang pengguna email jadi lebih mudah. Kita jadi tak perlu sering menandai sebuah email jika itu adalah spam.

Setelah fase itu mulai muncul ratusan implementasi dari ML yang kita gunakan sehari-hari saat ini. Merentang dari rekomendasi video di Youtube, fitur pengenalan wajah pada gambar, kontrol suara seperti pada Google Assistant, hingga sistem pemilihan pengemudi dan rekomendasi restoran pada aplikasi ojek online. Yup, itu semua adalah bentuk implementasi dari machine learning. Ternyata dekat ya, machine learning dengan kehidupan kita sehari-hari?

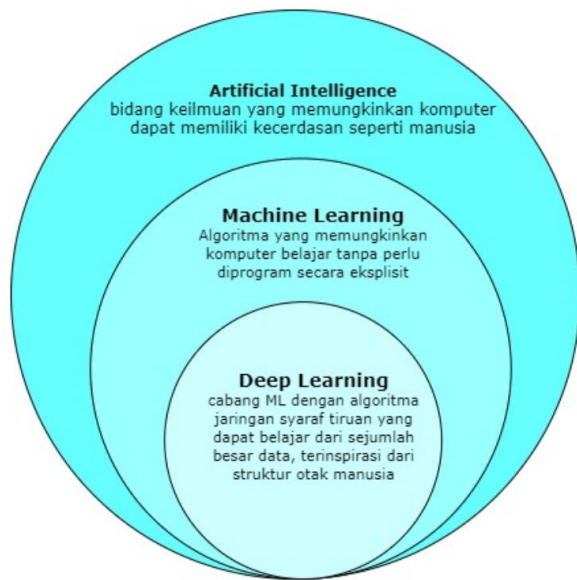
Selain machine learning, Anda juga pasti pernah mendengar tentang *artificial intelligence* dan *deep learning*. Lalu, apa hubungan antara AI, ML, dan deep learning?

Machine learning adalah cabang dari artificial intelligence. Kecerdasan buatan memiliki pengertian yang sangat luas tapi secara umum dapat dipahami sebagai komputer dengan kecerdasan layaknya manusia. Sedangkan ML memiliki arti lebih spesifik yaitu menggunakan metode statistika untuk membuat komputer dapat mempelajari pola pada data tanpa perlu diprogram secara eksplisit.

Lebih lanjut, deep learning adalah cabang machine learning dengan algoritma jaringan syaraf tiruan yang dapat belajar dan beradaptasi terhadap sejumlah besar data. Algoritma jaringan syaraf tiruan pada deep learning terinspirasi dari struktur otak

manusia. Algoritma ini memungkinkan mesin untuk melihat pola dari data yang tidak terstruktur atau data yang fiturnya tidak dapat ditentukan secara langsung. Contohnya, data gambar, teks, audio, dan video.

Jadi, ketika Anda ditanya tentang hubungan antara artificial intelligence, machine learning, dan deep learning di kemudian hari, Anda bisa menjawabnya demikian. Seperti inilah ilustrasinya dalam sebuah diagram.



1.2 Mengapa Machine Learning

Kemampuan mengenali pola dari sejumlah besar data sekaligus beradaptasi terhadap data baru adalah hal yang paling menarik, kuat, menggugah dari machine learning. Sebab seperti kita tahu, pemrograman tradisional memiliki keterbatasan dengan sifatnya yang rigid, dengan sekumpulan aturan “if” dan “else” untuk memproses data atau menyesuaikan dengan masukan. Menurut Muller [3], penggunaan aturan untuk penentuan keputusan dengan cara ini memiliki dua kelemahan utama:

- Logika yang digunakan untuk membuat keputusan bersifat spesifik pada ranah dan masalah tertentu. Mengubah sedikit masalah, membuat kita mungkin perlu menulis keseluruhan sistem.
- Mendesain aturan untuk sistem memerlukan pemahaman yang mendalam tentang bagaimana suatu keputusan harus dibuat oleh seorang ahli.

Padahal permasalahan terus muncul seiring waktu serta arus informasi dan data pun terus berubah. Machine learning, dengan kemampuannya yang secara alami bersifat adaptif terhadap data dan masukan baru, menawarkan solusi untuk masalah tersebut. Sebagai gambaran, berikut adalah contoh dua kategori permasalahan yang dapat diselesaikan dengan baik oleh algoritma machine learning.

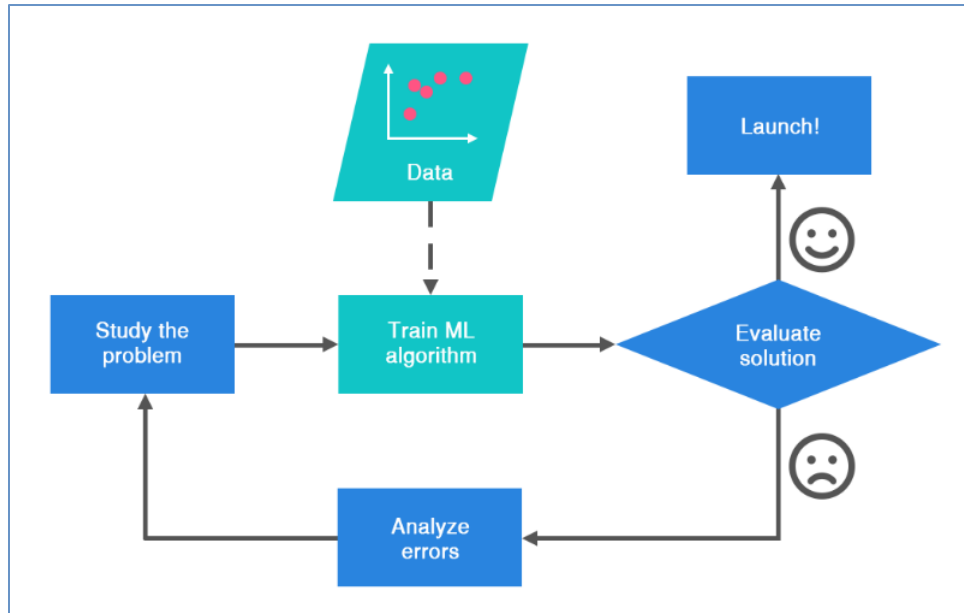
Masalah yang Solusinya Membutuhkan Banyak Penyesuaian dan Aturan

Bayangkan jika kita bertugas mengembangkan sebuah aplikasi filter spam dengan pemrograman tradisional. Langkah-langkah konvensional yang perlu kita lakukan adalah sebagai berikut.

Pertama, kita akan mendefinisikan bagaimana sebuah email termasuk kategori spam atau bukan. Misalnya, kita mengidentifikasi bahwa pada email spam umumnya terdapat kata-kata seperti “kaya”, “instan”, dan “murah”. Kemudian, kita menulis algoritma untuk setiap pola yang kita temukan pada email spam. Program pun akan menandai sebuah email spam jika menemui pola terkait. Terakhir, kita akan mengulangi kedua langkah tadi sampai program kita cukup baik untuk diluncurkan.

Karena kita menulis program menggunakan cara tradisional, hasilnya tentu daftar panjang berisi aturan-aturan rumit yang sulit untuk di-*maintain*, atau harus di-*update* secara berkala saat kita menemukan kosakata dan pola baru yang terkait dengan email spam.

Mari bandingkan jika kita menggunakan ML untuk mengembangkan filter spam tersebut. ML akan secara otomatis mempelajari pola kata-kata yang menentukan sebuah email spam atau bukan. Program dengan ML pun menjadi relatif lebih sederhana dan mudah untuk dipelihara. Seperti digambarkan oleh Geron [2], *flowchart* di bawah menunjukkan bagaimana alur pengembangan sebuah proyek Machine Learning.



Masalah Rumit yang Tidak Bisa Diselesaikan dengan Pemrograman Tradisional

Ada beberapa permasalahan yang sampai saat ini belum bisa dipecahkan dengan pendekatan pemrograman tradisional. Penyebabnya bisa jadi karena masalah tersebut terlalu rumit atau bisa juga karena belum diketahui algoritma pemrograman yang tepat untuk kasus tersebut.

Misal, kita ingin membangun sebuah sistem pengenalan suara. Menurut Geron [2], tahapan pertama yang perlu dilakukan adalah dengan membuat algoritma sederhana yang mampu membedakan kata “bagus” dan “indah”. Perhatikan bahwa kata “bagus” dimulai dengan suara bernada tinggi (“b”). Sebaliknya, kata “indah” dimulai dengan suara bernada rendah. Sehingga kita dapat membuat program untuk algoritma yang mengukur intensitas suara, kemudian menggunakannya untuk membedakan kedua kata tersebut.

Pertanyaannya, dapatkah hal ini kita terapkan pada ribuan jenis kata, yang diucapkan oleh jutaan orang, dalam berbagai macam bahasa, di lingkungan yang mungkin terdistraksi oleh suara-suara lainnya? Sangat rumit, bukan? Maka solusi terbaik yang dapat kita lakukan saat ini adalah, membuat algoritma yang dapat belajar dengan sendirinya melalui banyak data berupa sampel rekaman untuk setiap kata.

Tentu Anda dapat menerka, algoritma apa yang dapat menyelesaikan permasalahan ini? Betul, machine learning!

1.3 Jenis-jenis Machine Learning

Machine learning dibagi menjadi beberapa kategori. Tepatnya ada empat kategori besar, yaitu *supervised learning*, *unsupervised learning*, *semi-supervised learning*, dan *reinforcement learning*. Supervised learning dan unsupervised learning adalah dua kategori yang mungkin familiar bagi Anda. Tapi, apakah Anda tahu, pembagian kategori ini berdasarkan apa? Yup, berdasarkan karakteristik data dan jenis supervisi yang didapatkan oleh program selama pelatihan. Apa maksudnya? Simak pembahasan berikut ya.

Supervised Learning

Supervised learning adalah kategori machine learning yang menyertakan solusi yang diinginkan -yang disebut label- dalam proses pembelajarannya. Dataset yang digunakan telah memiliki label dan algoritma kemudian mempelajari pola dari pasangan data dan label tersebut. Algoritma supervised learning mudah dipahami dan performa akurasi pun mudah diukur. Supervised learning dapat dilihat sebagai sebuah mesin/robot yang belajar menjawab pertanyaan sesuai dengan jawaban yang telah disediakan manusia.

Unsupervised Learning

Anda mungkin sudah dapat mengira bahwa pada unsupervised learning, dataset yang digunakan tidak memiliki label. Betul, model unsupervised learning melakukan proses belajar sendiri untuk melabeli atau mengelompokkan data. Unsupervised learning dapat dilihat sebagai robot/mesin yang berusaha belajar menjawab pertanyaan secara mandiri tanpa ada jawaban yang disediakan manusia.

Semi-supervised Learning

Ini merupakan gabungan dari supervised learning dan unsupervised learning. Di sini dataset untuk pelatihan sebagian memiliki label dan sebagian tidak. Google Photos adalah contoh implementasi yang sering kita gunakan. Pada Google Photos kita bisa memberi *tag* atau label untuk setiap orang yang ada dalam sebuah foto. Alhasil, ketika kita mengunggah foto baru dengan wajah orang yang sebelumnya sudah kita beri label, Google Photos akan secara otomatis mengenali orang tersebut.

Salah satu contoh dari model semi supervised learning adalah *Deep Belief Network* (DBNs). DBNs adalah model grafis dengan multipel layer yang dapat belajar teknik mengekstrak data training secara efisien. Dua jenis layer pada DBNs adalah *visible* atau input layer dan *hidden layer*.

Menurut Geron [4], DBNs berdasar pada komponen unsupervised yang disebut restricted Boltzmann machine (RBMs). RBMs dilatih secara berurutan dengan algoritma unsupervised learning, kemudian seluruh sistem disesuaikan dengan teknik supervised learning.

Campbell [2] dalam tulisannya menyatakan bahwa pendekatan DBNs telah berhasil menyelesaikan pemodelan akustik pada speech recognition. DBNs menunjukkan sifat perkiraan yang kuat, peningkatan kinerja, dan merupakan parameter yang efisien.

Reinforcement Learning

Reinforcement Learning dikenal sebagai model yang belajar menggunakan sistem *reward* dan penalti. Menurut Winder [5], reinforcement learning adalah teknik yang mempelajari bagaimana membuat keputusan terbaik, secara berurutan, untuk memaksimalkan ukuran sukses kehidupan nyata. Entitas pembuat keputusan belajar melalui proses *trial* dan eror.

Reinforcement learning memiliki empat komponen, yaitu *action*, *agent*, *environment*, dan *reward*. Action adalah setiap keputusan yang diambil. Misal, saat kita berkendara, action yang kita lakukan adalah mengendalikan kemudi, menginjak gas, dan mengerem. Agent adalah entitas yang membuat keputusan, contohnya adalah perangkat lunak, atau robot, atau bahkan manusia. Environment adalah sarana untuk berinteraksi, yang dapat menerima action dan memberikan respon berupa hasil maupun data berupa satu set observasi baru. Reward diberikan saat agent berhasil menyelesaikan tantangan. Mekanisme feedback ini membuat agent belajar tentang tindakan mana yang menyebabkan kesuksesan (menghasilkan reward), atau kegagalan (menghasilkan penalti). Keempat komponen tersebut merepresentasikan Markov decision process (MDP).

Model reinforcement learning belajar agar terus mendapatkan reward dan menghindari penalti. [AlphaGo](#), sebuah program yang dikembangkan oleh Google DeepMind adalah contoh terkenal dari reinforcement learning. AlphaGo dibuat untuk memainkan permainan Go, sebuah permainan papan kuno yang berasal dari Cina. AlphaGo mempelajari setiap langkah dalam jutaan permainan Go, untuk terus mendapatkan reward yaitu memenangkan permainan. AlphaGo terkenal setelah menjadi program komputer pertama yang berhasil mengalahkan seorang pemain Go profesional yang juga merupakan juara dunia. Luar biasa, ya?

Penjelasan lebih lanjut tentang reinforcement learning akan kita pelajari di Kelas selanjutnya: Belajar Pengembangan Machine Learning. Pada kelas Belajar Machine Learning Pemula ini, kita akan fokuskan belajar tentang supervised learning dan unsupervised learning. Penjelasan lebih lanjut tentang algoritma-algoritma pada dua kategori ini akan tersedia di modul-modul selanjutnya. Semangat untuk lanjut ke materi berikutnya!

1.4 Tools dan Library Populer pada Python untuk Machine Learning dan Data Science

Python merupakan bahasa paling populer yang digunakan oleh para *Data Scientist* dan pengembang Machine Learning (ML). Python adalah kombinasi antara *general-purpose programming language* yang powerful dan *domain-specific scripting language* yang mudah digunakan. Mengapa Python sangat populer? Salah satu alasan yang menarik adalah beberapa perusahaan teknologi raksasa seperti Google dan Facebook memilih Python sebagai bahasa utama untuk *framework* machine learning mereka yaitu, TensorFlow dan PyTorch.

Keunggulan lain yang dimiliki Python adalah ia merupakan salah satu bahasa pemrograman yang mudah dipelajari karena sintaksnya sederhana. Banyak orang tanpa latar belakang IT namun ingin mengejar karir bidang ML atau *data science*, dapat dengan mudah mempelajari bahasa pemrograman Python. Hal ini juga dipermudah dengan kemampuan Python dalam berinteraksi langsung dengan kode, baik melalui mode *interactive* dan *script (scripting)*.

Apa itu script dan interactive mode?

Sederhananya, script mode adalah mode penulisan kode dalam sebuah berkas teks (umumnya berekstensi .py). Pada mode ini, berkas teks berisi kode kemudian dieksekusi oleh *compiler* atau *interpreter*. Mode script sangat cocok untuk menulis kode yang kompleks dan panjang. Mode ini lebih disukai oleh para *expert* yang telah berkecimpung lama di bidang pemrograman Python.

Sementara itu, mode interaktif pada Python memungkinkan kita untuk menulis dan menjalankan perintah secara langsung. Pada mode ini, kita dapat langsung mendapatkan output begitu perintah dieksekusi. Output kode dalam mode interaktif dipengaruhi oleh perintah terakhir yang kita jalankan. Mode ini biasanya digunakan untuk menulis baris kode yang pendek dan mencoba berbagai variasi sintaks. Ia cocok digunakan oleh para pemula yang sedang belajar pemrograman Python.

Tools untuk Pemrograman Python

Pada materi ini, kita akan mengenal beberapa tools mode interaktif yang sering digunakan dalam pemrograman Python. Tiga tools yang akan kita bahas di sini merupakan tools berbasis web (web-based interactive development environment) atau sering disebut Notebook. Notebook memiliki antarmuka yang fleksibel dan memungkinkan penggunaannya untuk menulis kode, menjalankannya, membuat konfigurasi, serta mengatur alur kerja dalam bidang data sains dan machine learning.

Apa saja tools yang dimaksud? Mari kita bahas sama-sama.

Jupyter Notebook

Jupyter Notebook merupakan perangkat lunak gratis, open-source, dan layanan web yang dapat digunakan untuk komputasi interaktif berbagai bahasa pemrograman, salah satunya Python. Instalasi Jupyter Notebook dapat dilakukan dengan beberapa cara. Silakan ikuti panduan di <https://jupyter.readthedocs.io/en/latest/install.html> untuk lebih detailnya. Selain itu, Anda juga dapat menggunakan Jupyter Notebook melalui browser dengan membuka tautan berikut <https://jupyter.org/try>, kemudian pilih Jupyter Notebook.

Google Colaboratory

Google Colaboratory atau sering disingkat Colab merupakan aplikasi yang memungkinkan kita untuk menulis dan mengeksekusi kode Python melalui browser. Ia sangat cocok digunakan untuk machine learning dan analisis data, serta sering digunakan oleh pemula. Untuk mulai menggunakan Google Colab, Anda dapat langsung mengunjungi tautan berikut: <https://colab.research.google.com/notebooks/>

IBM Watson Studio

IBM Watson Studio merupakan salah satu layanan dari IBM yang banyak digunakan oleh analis dan ilmuwan data. Anda juga dapat menjalankan kode secara online pada layanan seperti IBM Watson Studio tanpa perlu meng-*install* perangkat lunak apapun pada komputer. Sebelum menggunakan IBM Watson Studio, buatlah akun IBM Cloud terlebih dahulu. Akun IBM Cloud dapat dipakai untuk mengakses IBM Watson Studio, IBM Watson Machine Learning, dan IBM Cloud.

Lakukan beberapa hal berikut untuk mengakses IBM Watson Studio:

1. Buatlah akun pada IBM Cloud dengan mengunjungi [tautan ini](#) kemudian lakukan registrasi menggunakan email Anda.
2. Selanjutnya, setelah akun Anda jadi, login ke IBM Cloud dengan mengunjungi tautan <https://cloud.ibm.com/login>. Isi kolom IBMid dengan email yang telah Anda daftarkan di tahap sebelumnya.
3. Setelah login ke akun IBMCloud, ketiklah Object Storage pada *search bar*. Pada laman *Object Storage* pilih Lite pada bagian *Plan*. Perhatikan bahwa satu akun hanya dapat memiliki 1 Object Storage bertipe Lite. Jika Anda telah membuat object storage bertipe Lite sebelumnya, Anda harus menghapus dahulu object storage tersebut untuk bisa membuat object storage lite baru.
4. Langkah berikutnya adalah login ke IBM Watson Studio menggunakan akun IBM Cloud Anda melalui [tautan berikut](#).
5. Terakhir, buatlah Project di IBM Watson Studio dan tambahkan Asset baru (Jupyter Notebook Editor) dalam project Anda.

Selamat! Anda kini dapat menggunakan layanan Jupyter Notebook pada IBM Watson Studio.

Library Populer untuk Machine Learning dan Data Science

Faktor lain yang membuat Python populer adalah lengkapnya library yang dapat dipakai pada pengembangan proyek ML dari awal sampai akhir. Python memiliki library untuk *data loading*, *visualization*, *statistics*, *data processing*, *natural language processing*, *image processing*, dan lain sebagainya.

Beberapa library yang memudahkan kita dalam mengerjakan proyek ML ditunjukkan dalam daftar berikut ini.

Numpy



[Numpy](#) sangat terkenal sebagai library untuk memproses larik atau array. Fungsi-fungsi kompleks di baliknya membuat Numpy sangat tangguh dalam memproses larik multidimensi dan matriks berukuran besar. Library ML seperti TensorFlow juga menggunakan Numpy untuk memproses tensor atau sebuah larik N dimensi.

Pandas



[Pandas](#) menjadi salah satu library favorit untuk analisis dan manipulasi data. Kenapa keduanya penting? Sebelum masuk ke tahap pengembangan model, data perlu diproses dan dibersihkan. Proses ini bahkan merupakan proses yang paling banyak memakan

waktu dalam pengembangan proyek ML. Library pandas membuat pemrosesan dan pembersihan data menjadi lebih mudah.

Matplotlib



[Matplotlib](#) adalah sebuah library untuk membuat plot atau visualisasi data dalam 2 dimensi. Matplotlib mampu menghasilkan grafik dengan kualitas tinggi. Matplotlib dapat dipakai untuk membuat plot seperti histogram, scatter plot, grafik batang, pie chart, hanya dengan beberapa baris kode. Library ini sangat ramah pengguna.

Scikit Learn



[Scikit Learn](#) merupakan salah satu library ML yang sangat populer. Scikit Learn menyediakan banyak pilihan algoritma machine learning yang dapat langsung dipakai seperti klasifikasi, regresi, clustering, dimensionality reduction, dan pemrosesan data. Selain itu Scikit Learn juga dapat dipakai untuk analisis data.

TensorFlow



TensorFlow

[TensorFlow](#) adalah framework open source untuk machine learning yang dikembangkan dan digunakan oleh Google. TensorFlow memudahkan pembuatan model ML bagi pemula maupun ahli. Ia dapat dipakai untuk deep learning, computer vision, pemrosesan bahasa alami (Natural Language Processing), serta reinforcement learning.

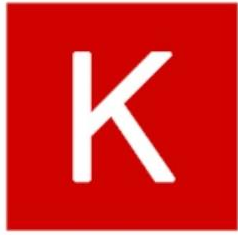
PyTorch



PyTorch

Dikembangkan oleh Facebook, [PyTorch](#) adalah library yang dapat dipakai untuk masalah ML, computer vision, hingga pemrosesan bahasa alami. Bersaing dengan TensorFlow khususnya sebagai framework machine learning, PyTorch lebih populer di kalangan akademisi dibanding TensorFlow. Namun dalam industri, TensorFlow lebih populer karena skalabilitasnya lebih baik dibanding PyTorch.

Keras



Keras

[Keras](#) adalah library deep learning yang luar biasa. Salah satu faktor yang membuat Keras sangat populer adalah penggunaannya yang minimalis dan simpel dalam mengembangkan deep learning. Keras dibangun di atas TensorFlow yang menjadikan Keras sebagai API dengan level lebih tinggi (Higher level API) dari TensorFlow sehingga antarmukanya lebih mudah dari TensorFlow. Keras sangat cocok untuk mengembangkan model deep learning dengan waktu yang lebih singkat atau untuk pembuatan prototipe.

1.5 Data Collecting

Pada materi sebelumnya Anda telah belajar tentang pengenalan machine learning (ML) dan jenis-jenis library machine learning. Apakah Anda bingung harus melangkah dari manakah untuk memulai proyek ML?

OK Jangan khawatir. Kita akan belajar sedikit demi sedikit ya. Tahap pertama dari proses pengerjaan proyek ML adalah data collecting, yaitu proses pengumpulan data.

“Data is the new oil. It’s valuable, but if unrefined it cannot really be used. It has to be changed into gas, plastic, chemicals, etc to create a valuable entity that drives profitable activity; so must data be broken down, analyzed for it to have value.”

-Clive Humby, 2006-

Kutipan di atas adalah kalimat terkenal tentang data yang pertama kali disampaikan oleh Clive Humby, seorang matematikawan asal Inggris pada tahun 2006. Kutipan tersebut menjadi sangat populer setelah [The Economist](#) mempublikasikan laporan tahun 2017 yang berjudul *The World’s most valuable resource is no longer oil, but data*.



Perangkat cerdas dan internet telah membuat data menjadi berlimpah. Banjir arus data yang terjadi di era digital mengubah sifat persaingan. Perusahaan teknologi raksasa berlomba-lomba mengumpulkan banyak data untuk meningkatkan produknya, menarik lebih banyak pengguna, menghasilkan lebih banyak data, dan seterusnya. Mereka menjangkau seluruh sektor ekonomi: Google bisa melihat apa yang ditelusuri dan dicari oleh orang-orang, Facebook bisa melihat apa yang mereka bagikan, dan Amazon

mengetahui apa yang mereka beli. Mereka seolah memiliki “God’s eyes view” tentang aktivitas di pasar mereka sendiri dan sekitarnya.

Luar biasa ya? Sekarang hampir semua perusahaan mengumpulkan data untuk sumber daya mereka.

Lantas, bagaimana cara mengumpulkan data? Ada tiga cara yang bisa kita lakukan untuk mengumpulkan data, yaitu.

- Mengekstraksi data (misal dari internet, riset, survei, dll).
- Mengumpulkan dan membuat dataset Anda sendiri dari nol.
- Menggunakan dataset yang telah ada.

Untuk saat ini, kita akan menggunakan dataset yang sudah ada dari platform penyedia data. Di masa mendatang tentu Anda dapat mencoba mengekstrak atau mengumpulkan dataset Anda sendiri ya.

Menemukan dataset yang tepat adalah salah satu langkah penting dalam proyek machine learning. Saat ini, tersedia banyak sumber data di internet yang dapat kita manfaatkan. Beberapa di antaranya yang perlu Anda ketahui adalah sebagai berikut.

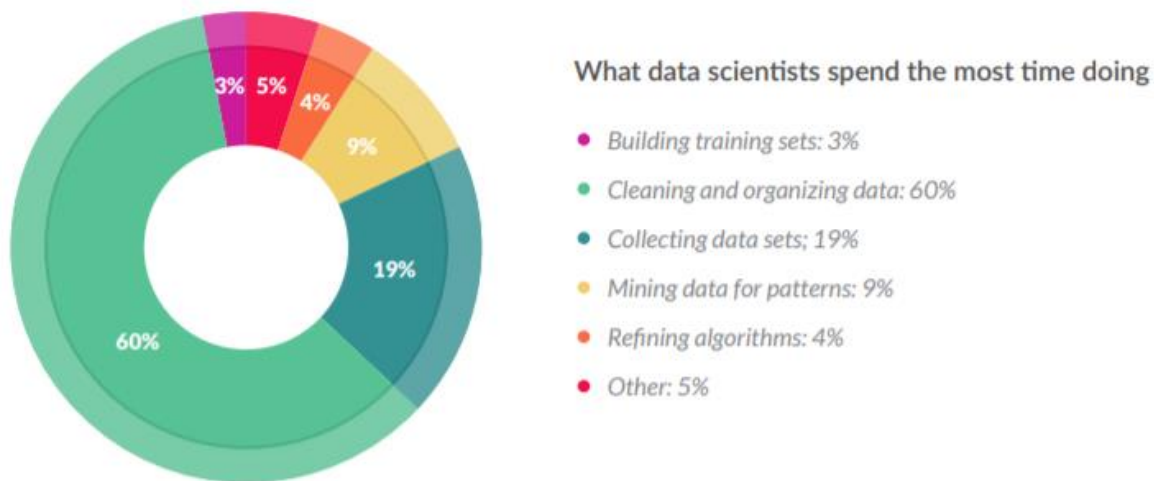
- UC Irvine Machine Learning Repository
[UCI ML Repository](#) adalah kumpulan database, teori, dan generator data yang digunakan oleh komunitas ML untuk analisis algoritma machine learning. Arsip tersebut awalnya dibuat sebagai arsip ftp pada tahun 1987 oleh David Aha, seorang mahasiswa pascasarjana UC Irvine. Sejak saat itu database UCI ML Repository ini digunakan secara luas oleh mahasiswa, staf pengajar, dan peneliti di seluruh dunia sebagai salah satu sumber utama dataset machine learning.
- Kaggle Dataset
[Kaggle](#) adalah komunitas belajar ilmu data paling populer di dunia. Kaggle memiliki peralatan dan sumber daya yang kuat untuk membantu kita belajar data science dan machine learning. Saat ini Kaggle memiliki 50.000 lebih publik dataset, baik dataset bersifat dummy ataupun riil yang dapat Anda unduh secara bebas.
- Google Dataset Search Engine
Pada akhir tahun 2018 Google meluncurkan [Dataset Search](#), sebuah mesin pencari dataset. Tools ini bertujuan untuk menyatukan ribuan repositori dataset yang berbeda agar dataset tersebut lebih mudah ditemukan oleh pengguna.
- Tensorflow Dataset
Seperti yang telah dijelaskan pada sub-modul sebelumnya, [TensorFlow](#) adalah framework open source untuk machine learning yang dikembangkan dan digunakan oleh Google. Selain menyediakan [learning resources](#), tensorflow juga menyediakan [data resources](#) yang cukup lengkap di library-nya mulai dari audio

data, images, text, video, dan lainnya.

- **US Government Data**
Bagi Anda yang tertarik untuk mempelajari fenomena yang terjadi di Amerika Serikat, pemerintah Amerika meluncurkan [data online resources](#) yang mudah diakses oleh publik. Isinya antara lain data badai, data angka kelulusan dan dropout, data hewan-hewan yang terancam punah, statistik kriminal, dan berbagai data menarik lainnya.
- **Satu Data Indonesia**
Pemerintah Indonesia, melalui portal resmi [Satu Data Indonesia](#) menjalankan kebijakan tata kelola data pemerintah yang bertujuan untuk menciptakan data berkualitas, mudah diakses, dapat dibagi, dan digunakan oleh Instansi Pusat serta Daerah. Data dalam portal ini dapat diakses secara terbuka dan dikategorikan sebagai data publik, sehingga tidak memuat rahasia negara, rahasia pribadi, atau hal lain sejenisnya sebagaimana diatur dalam Undang-undang nomor 14 Tahun 2008 tentang Keterbukaan Informasi Publik.
- **Open Data Pemerintah Jawa Barat**
[Open data Jawa Barat](#) adalah portal resmi data terbuka milik Pemerintah Provinsi Jawa Barat yang berisikan data-data dari Perangkat Daerah di lingkungan Pemerintah Provinsi Jawa Barat. Open Data Jawa Barat ada untuk dapat memenuhi kebutuhan data publik bagi masyarakat. Data disajikan dengan akurat, akuntabel, valid, mudah diakses dan berkelanjutan.

1.6 Data Cleaning

Kita mungkin berpikir pekerjaan data scientist atau machine learning engineer adalah membuat algoritma, mengeksplor data, membuat analisis, dan prediksi. Padahal faktanya, seseorang yang bekerja di bidang data membutuhkan sebagian besar waktunya untuk melakukan proses data cleaning. Hasil penelitian CrowdFlower dalam 2016 Data Science Report menyatakan bahwa 3 dari 5 data scientist yang disurvei menggunakan sebagian besar waktunya untuk membersihkan dan mengatur data.



Mengapa data cleaning begitu penting sehingga sebagian besar waktu digunakan untuk menyelesaikan proses ini?

Data cleaning penting sebab proses ini meningkatkan kualitas data yang juga berpengaruh terhadap produktivitas kerja secara keseluruhan. Data yang tidak akurat bisa berpengaruh buruk terhadap akurasi dan performa model. Saat kita melakukan proses data cleaning, kita membuang data dan informasi yang tidak dibutuhkan sehingga kita akan mendapatkan data yang berkualitas. Data yang akurat dan berkualitas akan berpengaruh positif terhadap proses pengambilan keputusan. Pernahkah mendengar ungkapan “Garbage In - Garbage Out?” Dalam konteks machine learning, jika input yang Anda masukkan itu buruk, sudah barang tentu hasil olah data Anda pun akan buruk.

Sampai sini paham, ya?

Data cleaning merupakan tahapan penting yang tidak boleh Anda lewatkan. Berikut adalah beberapa hal umum yang harus diperhatikan dalam proses data cleaning:

1. Konsistensi Format

Sebuah variabel mungkin tidak memiliki format yang konsisten seperti penulisan tanggal 10-Okt-2020 versus 10/10/20. Format jam yang berbeda seperti 17.10

versus 5.10 pm. Penulisan uang seperti 17000 versus Rp 17.000. Data dengan format berbeda tidak akan bisa diolah oleh model machine learning. Solusinya, format data harus disamakan dan dibuat konsisten terlebih dahulu.

2. Skala Data

Jika sebuah variabel memiliki jangka dari 1 sampai 100, pastikan tidak ada data yang lebih dari 100. Untuk data numerik, jika sebuah variabel merupakan bilangan positif, maka pastikan tidak ada bilangan negatif.


3. Duplikasi data

Data yang memiliki duplikat akan mempengaruhi model machine learning, apalagi jika data duplikat tersebut besar jumlahnya. Untuk itu kita harus memastikan tidak ada data yang terduplikasi.

4. Missing Value

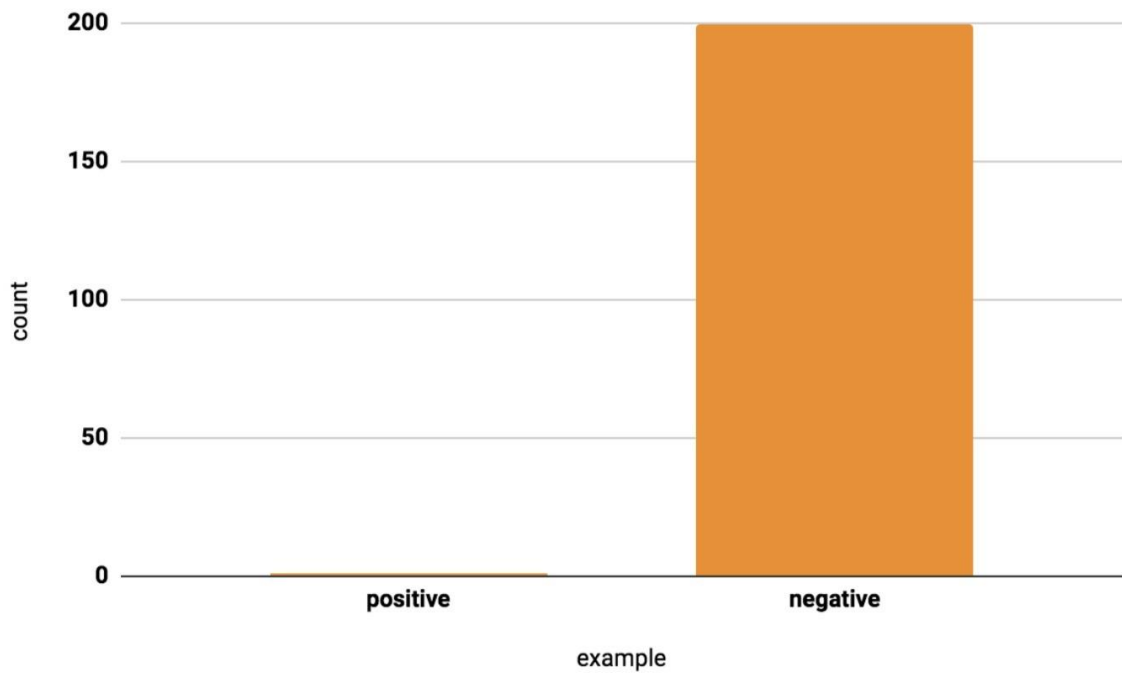
Missing value terjadi ketika data dari sebuah record tidak lengkap. Missing value sangat mempengaruhi performa model machine learning. Ada 2 (dua) opsi untuk mengatasi missing value, yaitu menghilangkan data missing value atau mengganti nilai yang hilang dengan nilai lain, seperti rata-rata dari kolom tersebut (mean) atau nilai yang paling sering muncul (modus), atau nilai tengah (median).

Berikut adalah contoh handling missing value dengan nilai rata-rata (mean).

kolom 1	kolom 2	kolom 3		kolom 1	kolom 2	kolom 3
8	NaN	NaN	mean()	8.0	6.0	4.0
6	3.0	4.0		6.0	3.0	4.0
9	9.0	NaN		9.0	9.0	4.0

5. Skewness Distribution

Skewness adalah kondisi di mana dataset cenderung memiliki distribusi data yang tidak seimbang. Skewness akan mempengaruhi data dengan menciptakan bias terhadap model. Apa itu bias? Sebuah model cenderung memprediksi sesuatu karena ia lebih sering mempelajari hal tersebut. Misalkan ada sebuah model untuk pengenalan buah di mana jumlah jeruk 92 buah dan apel 8 buah. Distribusi yang tidak imbang ini akan mengakibatkan model lebih cenderung memprediksi jeruk daripada apel.



Cara paling simpel untuk mengatasi skewness adalah dengan menyamakan proporsi kelas mayoritas dengan kelas minoritas. Untuk teknik lebih lanjut dalam mengatasi skewness atau imbalance data, Anda bisa membacanya di [tautan ini](#). Jika Anda ingin membaca lebih lanjut tentang proses data cleaning, silakan buka [tautan ini](#) atau [tautan berikut](#).

1.7 Data Processing

Pada tahap ini, setelah data diambil dari sumber tertentu, ia dimasukkan pada suatu environment. Lantas data diproses agar bisa diolah oleh model machine learning. Menjalankan proses machine learning sama seperti mengajari seorang anak kecil. Ketika mengajari anak kecil membedakan antara buah apel dan buah jeruk, kita tinggal memperlihatkan buahnya dan memberi tahu mana apel dan mana jeruk. Namun demikian, komputer saat ini belum secanggih dan seintuitif itu sehingga kita perlu mempersiapkan data dengan data processing agar bisa dimengerti komputer.

Salah satu library yang paling populer untuk pengolahan data dalam machine learning adalah Pandas Library. Pandas Library adalah salah satu library yang wajib Anda kuasai dalam bidang machine learning.

Pandas Library

Pandas adalah sebuah library open source yang dipakai untuk menganalisis dan memanipulasi data. Pandas dibangun menggunakan bahasa pemrograman Python yang menawarkan struktur data dan operasi untuk manipulasi tabel numerik dan time series. Tabel numerik adalah tabel yang berisi bilangan numerik dan Tabel time series adalah tabel yang berubah seiring waktu, misalnya tabel yang memuat perubahan nilai pasar saham untuk setiap menitnya.

Berbagai jenis data yang umum dipakai dalam ML seperti CSV dan SQL dapat diubah menjadi dataframe pandas. Dataframe adalah sebuah tabel yang terdiri dari kolom dan baris dengan banyak tipe data di dalamnya. Pandas terintegrasi dengan library machine learning yang populer seperti Scikit Learn (SKLearn) dan Numpy.

Pandas mendukung banyak jenis data yang dapat dipakai dalam sebuah project machine learning. Berikut adalah beberapa contoh data yang dapat diolah dengan pandas.

- **CSV**
CSV adalah sebuah format data di mana elemen dari setiap baris dipisahkan dengan koma. CSV sendiri adalah singkatan dari Comma Separated Value.
- **SQL**
Structured Query Language adalah sebuah data yang berasal dari sebuah relational database. Format data ini berisi sebuah tabel yang memiliki format data seperti integer, string, float, dan biner.
- **EXCEL**
Excel adalah berkas yang didapat dari spreadsheet seperti Microsoft Excel atau Google Spreadsheet. File Excel biasanya memuat data numerik.

- **SPSS**
SPSS atau Statistical Package for the Social Science adalah sebuah berkas dari perangkat lunak yang biasa dipakai untuk statistik dan pengolahan data. Berkas SPSS yang disimpan memiliki ekstensi .sav.
- **JSON**
JSON atau Javascript Object Notation adalah salah satu format data yang menggunakan sistem Key - Value di mana sebuah nilai disimpan dengan key tertentu untuk memudahkan mencari data.

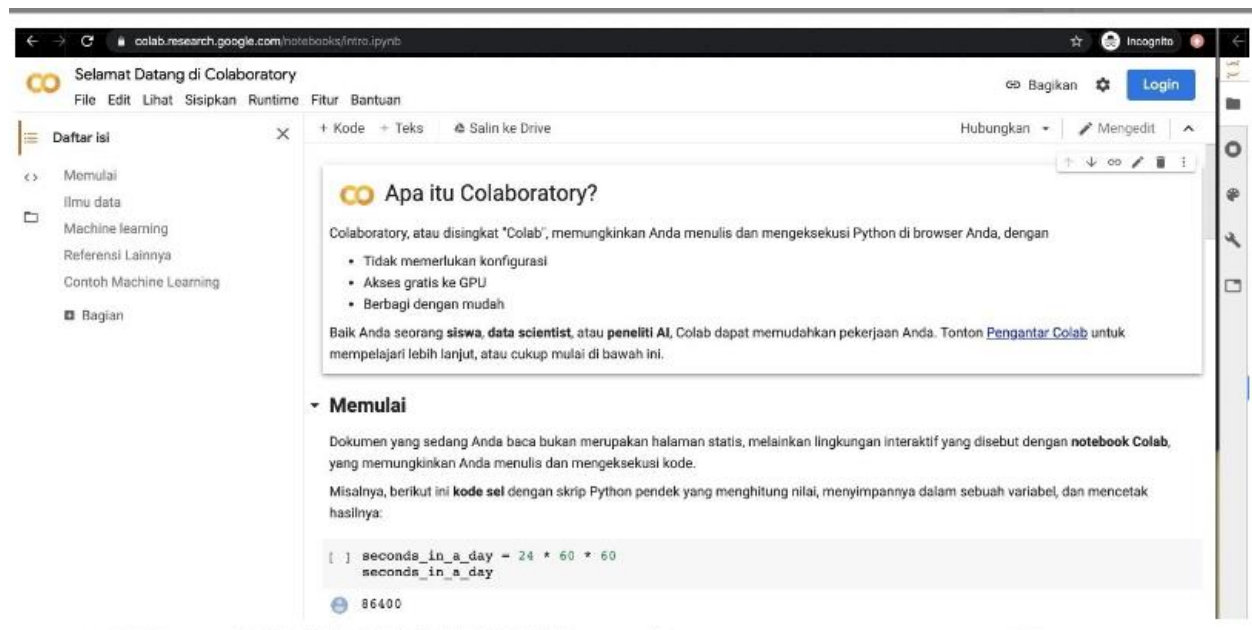
Pada kelas ini kita hanya akan menggunakan data berjenis csv karena data tipe ini mudah didapat di situs penyedia data seperti yang telah dibahas di sub-modul sebelumnya.

Untuk tahapan selanjutnya Anda dapat menulis dan menjalankan kode pada Google Colaboratory.

Google Colaboratory

Google Colaboratory atau sering juga disebut “Colab” adalah sebuah peranti dari Google yang dibuat untuk keperluan mengolah data, belajar, dan bereksperimen khususnya dalam bidang Machine Learning. Colab berjalan sepenuhnya pada Cloud dengan memanfaatkan media penyimpanan Google Drive.

Untuk membuka Google Colab, Anda dapat membuka tautan ini colab.research.google.com. Berikut adalah tampilan antarmuka dari Google Colaboratory. Untuk membuat dan menyimpan proyek pada Google Colab, Anda harus login terlebih dahulu menggunakan akun Google Anda.



The screenshot shows the Google Colaboratory interface in a web browser. The address bar displays `colab.research.google.com/notebooks/intro.ipynb`. The page title is "Selamat Datang di Colaboratory" (Welcome to Colaboratory). The left sidebar contains a "Daftar isi" (Table of contents) with links to "Memulai", "Ilmu data", "Machine learning", "Referensi Lainnya", "Contoh Machine Learning", and "Bagian". The main content area is titled "Apa itu Colaboratory?" and includes the following text:

Colaboratory, atau disingkat "Colab", memungkinkan Anda menulis dan mengeksekusi Python di browser Anda, dengan

- Tidak memerlukan konfigurasi
- Akses gratis ke GPU
- Berbagi dengan mudah

Baik Anda seorang **siswa**, **data scientist**, atau **peneliti AI**, Colab dapat memudahkan pekerjaan Anda. Tonton [Pengantar Colab](#) untuk mempelajari lebih lanjut, atau cukup mulai di bawah ini.

Memulai

Dokumen yang sedang Anda baca bukan merupakan halaman statis, melainkan lingkungan interaktif yang disebut dengan **notebook Colab**, yang memungkinkan Anda menulis dan mengeksekusi kode.

Misalnya, berikut ini **kode sel** dengan skrip Python pendek yang menghitung nilai, menyimpannya dalam sebuah variabel, dan mencetak hasilnya:

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day
```

86400

Untuk mengetahui lebih lanjut tentang Colab, kunjungi [tautan](#) berikut.

1.8 Latihan Konversi Pandas Dataframe

Kode program yang akan diajarkan di sini bisa diunduh di tautan [berikut ini](#). Untuk membukanya, upload berkasnya ke Google Colab.

Tujuan

Setelah sebelumnya kita mengenal library Pandas, sekarang kita akan belajar menggunakan library tersebut.

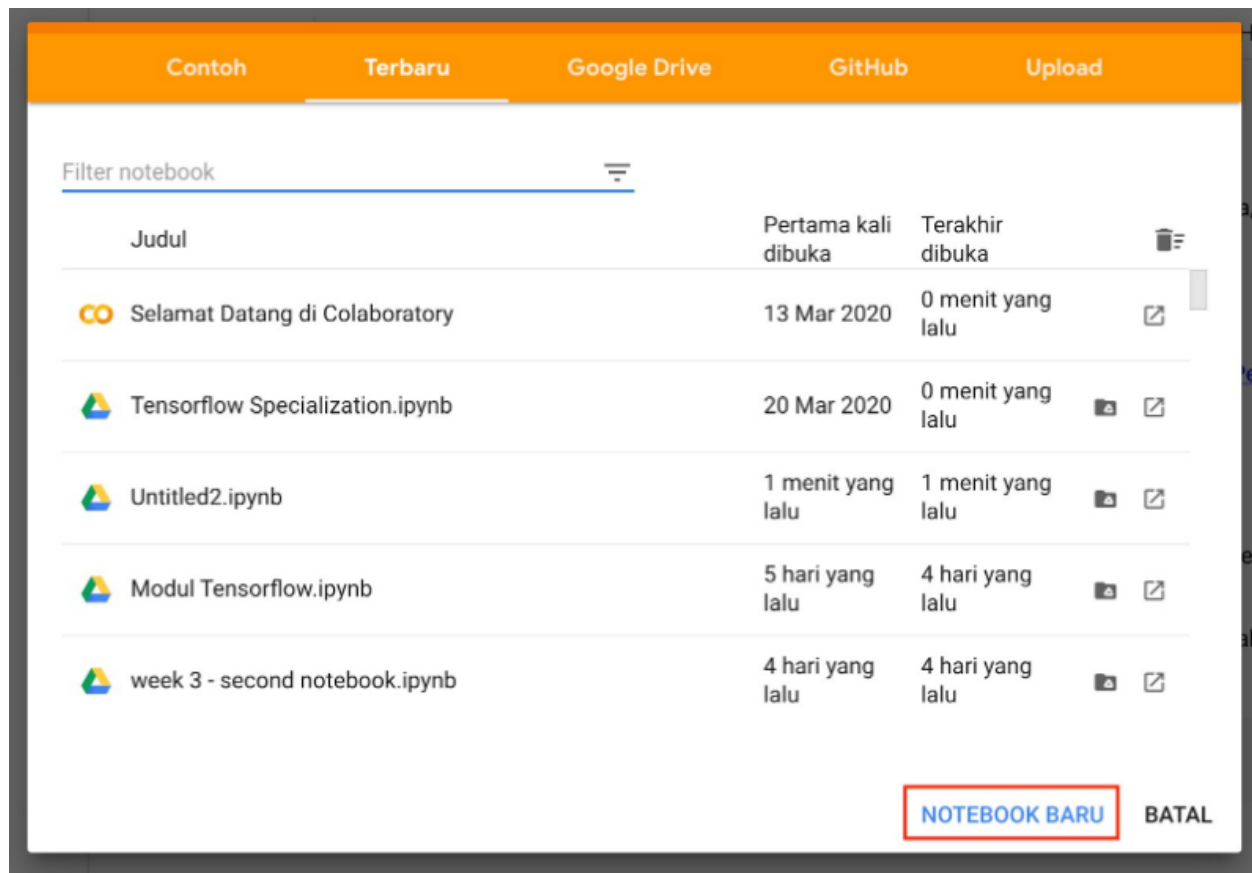
Tahapan Latihan

Pandas memiliki fungsi untuk mengubah data dari berbagai jenis menjadi *dataframe* seperti yang telah dibahas di submodul sebelumnya. Tahapan dalam mengubah data menjadi dataframe adalah sebagai berikut:

1. Persiapkan dataset yang akan dikonversi.
2. Impor library Pandas.
3. Gunakan fungsi `read_csv()` untuk mengonversi data.

Codelab

Pertama kita bisa buka Google Colab, dengan mengunjungi tautan colab.research.google.com. Anda bisa menekan tombol **Notebook Baru** pada halaman pertama.



Colab memiliki direktori yang menyimpan berkas-berkas umum yang dapat dipakai untuk berbagai keperluan. Berkas-berkas bawaan tersebut disimpan dalam sebuah direktori bernama **'sample_data'**.

Untungnya, notebook pada Colab kita memiliki berkas csv yang dapat kita pakai untuk mencoba *library pandas*. Anda juga bisa melihat beberapa berkas bawaan yang disediakan Colab dengan menggunakan **library os**.

1. `import os`
2. `os.listdir('sample_data')`

Pada notebook baru, kita akan melakukan **Import library pandas** dan memberikan alias **pd** untuk library tersebut. Memberikan alias *pd* terhadap library pandas adalah praktek yang umum dilakukan para pengembang ML.

1. `import pandas as pd`

Pada latihan ini kita akan mencoba melakukan konversi dari berkas '**california_housing_train.csv**' yang merupakan bawaan dari Colab. Untuk mengubah berkas csv menjadi *dataframe*, kita menggunakan fungsi **read_csv()** pada library pandas kemudian menyimpannya pada sebuah variabel yaitu df.

1. `df = pd.read_csv('sample_data/california_housing_train.csv')`
2. `df.head()`

Untuk melihat apakah konversi berhasil, Anda bisa memanggil fungsi **head()** pada dataframe. Lalu jalankan cell tersebut dengan klik tombol *run* di sebelah kiri. Fungsi **head()** akan menampilkan 5 baris teratas dari sebuah dataframe.

Jika keluaran dari cell seperti di bawah, maka selamat, Anda berhasil melakukan konversi.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.0	1.4936	66900.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.0	1.8200	80100.0
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.0	1.6509	85700.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.0	3.1917	73400.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.0	1.9250	65500.0

1.9 Data Preparation dengan Teknik One-Hot-Encoding

Setelah dataset dibersihkan, masih ada beberapa tahap yang perlu dilakukan agar dataset benar-benar siap untuk diproses oleh model machine learning. Biasanya, dataset Anda akan terdiri dari dua jenis data: kategorik dan numerik. Contoh data numerik adalah: ukuran panjang, suhu, nilai uang, hitungan dalam bentuk angka, dll, yang terdiri dari bilangan integer (seperti -1, 0, 1, 2, 3, dan seterusnya) atau bilangan float (seperti -1.0, 2.5, 39.99, dan seterusnya). Setiap nilai dari data dapat diasumsikan memiliki hubungan dengan data lain karena data numerik dapat dibandingkan dan memiliki ukuran yang jelas. Misal, Anda dapat mengatakan bahwa panjang 39 m lebih besar dibanding 21 m. Jenis data ini terdefinisi dengan baik, dapat dioperasikan dengan metode statistik, dan mudah dipahami oleh komputer.

Jenis data lain yang sering kita temui adalah data kategorik. Data kategorik adalah data yang berupa kategori dan berjenis string, tidak dapat diukur atau didefinisikan dengan angka atau bilangan. Contoh data kategorik adalah sebuah kolom pada dataset yang berisi perkiraan cuaca seperti cerah, berawan, hujan, atau berkabut. Contoh lain dari data kategorik adalah jenis buah misalnya apel, pisang, semangka, dan jeruk. Pada jenis data ini, kita tidak bisa mendefinisikan operasi perbandingan seperti lebih besar dari, sama dengan, dan lebih kecil dari. Dan dengan demikian, kita juga tidak dapat mengurutkan dan melakukan operasi statistik terhadap data jenis ini.

Umumnya, model machine learning tidak dapat mengolah data kategorik, sehingga kita perlu melakukan konversi data kategorik menjadi data numerik. Banyak model machine learning seperti Regresi Linear dan Support Vector Machine (kedua model ini akan dibahas pada modul-modul selanjutnya) yang hanya menerima input numerik sehingga tidak bisa memproses data kategorik. Salah satu teknik untuk mengubah data kategorik menjadi data numerik adalah dengan menggunakan One Hot Encoding atau yang juga dikenal sebagai dummy variables. One Hot Encoding mengubah data kategorik dengan membuat kolom baru untuk setiap kategori seperti gambar di bawah.

Gender		
	Laki-laki	Perempuan
Perempuan	0	1
Laki-laki	1	0
Perempuan	0	1
Perempuan	0	1
Laki-laki	1	0

Pada modul latihan nanti, Anda akan berlatih bagaimana menerapkan teknik one-hot-encoding pada dataset. Sudah siap untuk lanjut?

1.10 Data Preparation dengan Normalization dan Standardization

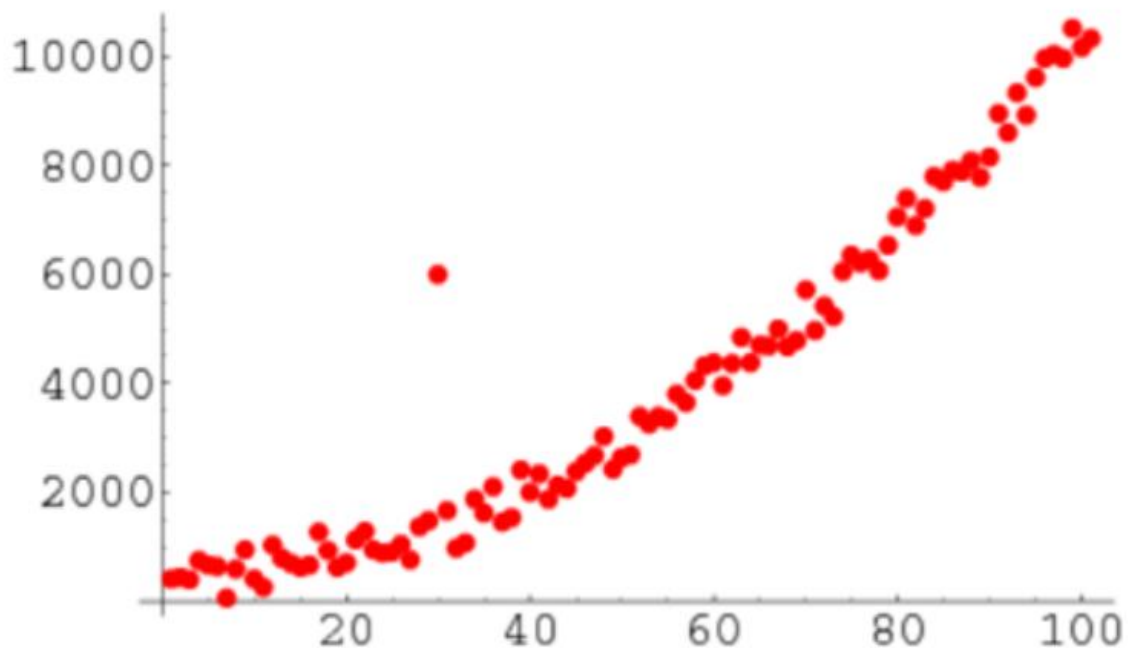
Selain konversi data kategorik menjadi numerik, ada beberapa teknik lain dalam *data preparation*. Teknik yang akan dibahas antara lain membuang *outlier*, *normalization*, dan *standardization*.

Outlier Removal

Dalam statistik, outlier adalah sebuah nilai yang jauh berbeda dari kumpulan nilai lainnya dan dapat mengacaukan hasil dari sebuah analisis statistik. Outlier dapat disebabkan oleh kesalahan dalam pengumpulan data atau nilai tersebut benar ada dan memang unik dari kumpulan nilai lainnya.

Apa pun alasan kemunculannya, Anda perlu tahu cara mengidentifikasi dan memproses outlier. Ini adalah bagian penting dalam persiapan data di dalam machine learning. Salah satu cara termudah untuk mengecek apakah terdapat outlier dalam data kita adalah dengan melakukan visualisasi.

Berikut adalah contoh visualisasi terhadap data yang memiliki outlier.



Dapat dilihat dengan jelas bahwa terdapat satu sampel yang jauh berbeda dengan sampel-sampel lainnya. Setelah mengetahui bahwa di data kita terdapat outlier, kita dapat mencari lalu menghapus sampel tersebut dari dataset.

Normalization

Normalization adalah salah satu teknik yang dipakai dalam data preparation. Tujuan dari normalisasi adalah mengubah nilai-nilai dari sebuah fitur ke dalam skala yang sama. Normalization memungkinkan kenaikan performa dan stabilitas dari sebuah model machine learning.

Nama	Gaji	Umur
A	12.000.000	33
B	35.000.000	45
C	4.000.000	23
D	6.500.000	26
E	9.000.000	29

Contoh dari normalization adalah ketika kita memiliki dataset seperti di atas yang memiliki fitur umur dengan skala 23 sampai 45 tahun dan fitur penghasilan dengan skala 4.000.000 sampai 35.000.000. Di sini kita melihat bahwa fitur penghasilan sekitar satu juta kali lebih besar dari fitur umur dan menunjukkan kedua fitur ini berada pada skala yang sangat jauh berbeda.

Ketika membangun model seperti regresi linear, fitur penghasilan akan sangat mempengaruhi prediksi dari model karena nilainya yang jauh lebih besar daripada umur, walaupun tidak berarti fitur tersebut jauh lebih penting dari fitur umur.

Salah satu contoh dari normalization adalah *min-max scaling* di mana nilai-nilai dipetakan ke dalam skala 0 sampai 1. SKLearn menyediakan library untuk normalization

Pada Colab kita **Import** library **MinMaxScaler** dan masukkan data dari tabel sebelumnya.

```
1. from sklearn.preprocessing import MinMaxScaler
```

```
2. data = [[12000000, 33], [35000000, 45], [4000000, 23], [6500000, 26], [9000000, 29]]
```

Pada cell selanjutnya kita buat sebuah objek `MinMaxScaler` dan panggil fungsi `fit()` dan mengisi argumen **data** seperti potongan kode di bawah. Fungsi `fit()` dari objek `MinMaxScaler` adalah fungsi untuk menghitung nilai minimum dan maksimum pada tiap kolom.

```
1. scaler = MinMaxScaler()
```

```
2. scaler.fit(data)
```

Apabila dijalankan, maka hasilnya sebagai berikut.

```
↳ MinMaxScaler()
```

Sampai pada fungsi `fit()` ini, komputer baru menghitung nilai minimum dan maksimum pada tiap kolom dan belum melakukan operasi scaler pada data. Terakhir kita panggil fungsi `transform()` yang akan mengaplikasikan scaler pada data, sebagai berikut.

```
1. print(scaler.transform(data))
```

Hasil dari kode di atas seperti ditunjukkan pada gambar berikut.

```
↳ [[0.25806452 0.45454545]
    [1.         1.         ]
    [0.         0.         ]
    [0.08064516 0.13636364]
    [0.16129032 0.27272727]]
```

Setiap nilai dari kolom gaji dan umur telah dipetakan pada skala yang sama seperti di bawah ini.

Nama	Gaji	Umur
A	0.25806452	0.45454545
B	1	1
C	0	0
D	0.08064516	0.13636364

Nama	Gaji	Umur
E	0.16129032	0.27272727

Untuk informasi lebih detail tentang Min Max Scaler, silakan kunjungi [tautan](#) berikut.

Standardization

Standardization adalah proses konversi nilai-nilai dari suatu fitur sehingga nilai-nilai tersebut memiliki skala yang sama. Z score adalah metode paling populer untuk standardisasi di mana setiap nilai pada sebuah atribut numerik akan dikurangi dengan rata-rata dan dibagi dengan standar deviasi dari seluruh nilai pada sebuah kolom atribut.

$$z = \frac{value - mean}{standard\ deviation}$$

Fungsi standardisasi itu serupa dengan normalization. Keduanya berfungsi menyamakan skala nilai dari tiap atribut pada data. SKLearn menyediakan library untuk mengaplikasikan standard scaler pada data.

Nama	Gaji	Umur
A	12.000.000	33
B	35.000.000	45
C	4.000.000	23
D	6.500.000	26
E	9.000.000	29

Pada colab di cell pertama kita akan mengimpor library **preprocessing** dari **scikit learn** lalu membuat data dummy sesuai dengan tabel di atas.

```
1. from sklearn import preprocessing

2. data = [[12000000, 33], [35000000, 45], [4000000, 23], [6500000, 26], [9000000, 29]]
```

Selanjutnya kita buat object scaler dan panggil fungsi fit dari scaler pada data. Fungsi fit memiliki fungsi untuk menghitung rata-rata dan deviasi standar dari setiap kolom atribut untuk kemudian dipakai pada fungsi transform.

```
1. scaler = preprocessing.StandardScaler().fit(data)
```

Terakhir, kita panggil fungsi transform untuk mengaplikasikan standard scaler pada data. Untuk melihat hasil dari standard scaler kita tinggal memanggil objek scaler yang telah kita buat sebelumnya. Kodenya sebagai berikut.

```
1. data = scaler.transform(data)
```

```
2. data
```

Hasil akhirnya apabila dijalankan seperti di bawah ini.

```
array([[ -0.11638732,  0.23521877],  
       [ 1.94277296,  1.80334389],  
       [-0.83261698, -1.07155217],  
       [-0.60879521, -0.67952089],  
       [-0.38497344, -0.28748961]])
```

Untuk informasi lebih detail tentang standardization, silakan kunjungi [tautan](#) berikut.

1.11 Data Storage/Warehouse

Data warehouse pertama kali muncul pada tahun akhir 1980-an. Tujuan awalnya adalah untuk membantu proses aliran data dari sistem operasional ke dalam sistem pendukung keputusan atau *decision-support system* (DSS). Seiring berjalannya waktu, data warehouse berkembang menjadi lebih efisien. Ia berevolusi dari penyimpanan informasi pendukung platform *business intelligence* menjadi infrastruktur analitis luas yang mendukung berbagai macam aplikasi.

AI dan machine learning telah mengubah banyak hal dari mulai industri, layanan, aset perusahaan, juga sistem data warehouse. Ekspansi big data dan penerapan teknologi digital mendorong perubahan dalam kapabilitas data warehouse. Untuk mendukung hal tersebut, ada beberapa *tools* yang perlu kita ketahui.

RDBMS

Dalam model relasional, sebuah database terdiri dari banyak tabel. Sebuah tabel dibentuk dari kolom dan baris yang memuat nilai tertentu. Konsep Relational Database Management System (RDBMS) sendiri merupakan sistem yang mendukung adanya hubungan atau relasi antar tabel pada suatu database. Setiap tabel dihubungkan dengan tabel lainnya dengan menggunakan primary key dan foreign key. Saat ini sudah banyak jenis database yang menerapkan model RDBMS. Sebut saja MySQL, PostgreSQL, dan Microsoft SQL Server.

NoSQL

Sesuai dengan namanya NoSQL adalah jenis basis data yang tidak menggunakan bahasa SQL dalam manipulasi datanya. Dalam penyimpanan datanya, NoSQL memiliki beberapa teknik penyimpanan yaitu: dokumen, graph, key-value, dan column based.

1. Dokumen : Menghubungkan setiap kunci dengan struktur data kompleks yang disebut dokumen.
2. Graph : Menyimpan informasi tentang jaringan data, seperti koneksi sosial.
3. Nilai-kunci : Database NoSQL paling sederhana di mana setiap elemen dalam database disimpan sebagai nilai yang diasosiasikan dengan sebuah kunci.
4. Kolom : Menyimpan data yang memiliki volume besar, di mana setiap elemen data disimpan pada kolom bukan pada baris.

Beberapa database NoSQL terpopuler adalah MongoDB, CouchDB, Cassandra, Redis, Neo4J, dan Riak. Jika ingin mengetahui lebih lanjut tentang NoSQL, kunjungi [tautan](#) berikut.

Firestore Realtime Database

Sesuai namanya, “Database Realtime” adalah database yang menyimpan data yang berubah seiring waktu. Data jumlah penjualan harian, pengunjung mall setiap jam, arus lalu lintas setiap menit, atau fluktuasi saham setiap detik merupakan beberapa contoh data realtime. Data pada database realtime disimpan dalam format waktu dan nilai pada waktu yang terkait seperti gambar di bawah.

Timestamp	Metric 1
2019-03-28 00:00:01	2356
2019-03-28 00:00:02	6874
2019-03-28 00:00:03	3245
2019-03-28 00:00:04	2340

Firestore Realtime Database (FRD) adalah database berbasis cloud yang didesain khusus untuk mengelola data realtime. FRD dapat menyimpan dan melakukan sinkronisasi data secara realtime di mana setiap kali ada perubahan data terbaru, FRD langsung menyimpannya pada Cloud. FRD juga dilengkapi fitur offline di mana ketika tidak ada koneksi internet, FRD akan menyimpan data secara lokal, kemudian saat *online*, akan melakukan sinkronisasi ke Cloud. Keren, kan?

Spark

Apache Spark adalah perangkat lunak untuk pemrosesan dan analisis data berskala besar. Spark dapat digunakan dalam proses ETL (Extract, Transform, Load), data streaming, perhitungan grafik, SQL, dan machine learning. Untuk machine learning, Spark menyediakan MLlib yang berisi implementasi model machine learning seperti klasifikasi, regresi, pengklasteran, penurunan dimensi, dan pemfilteran kolaboratif.

Big Query

BigQuery adalah *data warehouse* berbasis cloud untuk perusahaan yang menawarkan penyimpanan data berbasis SQL dan analisis data berukuran besar. Karena berbasis cloud dan tidak ada infrastruktur yang perlu dikelola, pengguna dapat berfokus pada pengolahan data tanpa memerlukan seorang *administrator database*.

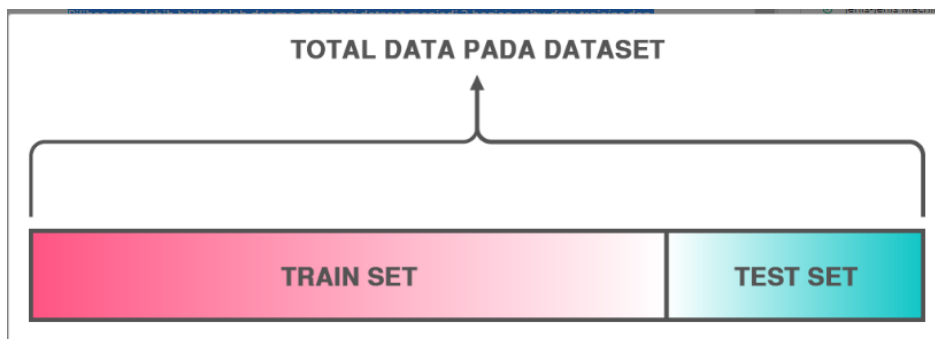
Sebagai ML Engineer masa depan, kita harus mampu mengoperasikan berbagai jenis data storage dan data warehouse. Sebabnya, perusahaan tempat kita ingin bekerja sebagai ML Engineer nanti, tidak selalu menggunakan *data warehouse* atau *data storage* yang sama.

1.12 Datasets

Dataset yang telah dibersihkan dan diproses kemudian siap kita latih dengan machine learning. Satu-satunya cara untuk mengetahui apakah model machine learning kita bagus atau tidak adalah dengan mengujinya pada kasus atau data baru yang belum dikenali oleh model. Kita bisa saja membuat model dan langsung mengujinya pada tahap produksi lalu memonitor kualitasnya. Tapi jika ternyata model yang kita kembangkan bekerja dengan buruk, pelanggan dan klien kita akan komplain. Selain itu, cara ini tentu akan memakan sumber daya dan biaya yang lebih besar.

Training Set dan Test Set

Pilihan yang lebih baik adalah dengan membagi dataset menjadi 2 bagian yaitu *data training* dan *data testing*. Berikut adalah gambaran bagaimana total data pada dataset dibagi menjadi dua bagian: *train set* dan *test set*.



Dengan demikian, kita bisa melakukan pelatihan model pada train set, kemudian mengujinya pada test set --sekumpulan data yang belum dikenali model. Ingat bahwa membandingkan hasil prediksi dengan label sebenarnya dalam test set merupakan proses evaluasi performa model. Dengan menguji model terhadap data testing, kita dapat melihat kesalahan yang dibuat dan memperbaikinya sebelum mulai membawa model kita ke tahap produksi.

Penting untuk kita memilih rasio yang sesuai dalam pembagian dataset. Saat membagi dataset, kita perlu membuat informasi pada kedua bagian tetap berimbang. Kita tidak ingin mengalokasikan terlalu banyak informasi pada data testing agar algoritma ML dapat belajar dengan baik pada data training. Tetapi, jika alokasi data pada data testing terlalu kecil, kita tidak bisa mendapatkan estimasi performa model yang akurat.

Data testing diambil dengan proporsi tertentu. Pada praktiknya, pembagian data training dan data testing yang paling umum adalah 80:20, 70:30, atau 60:40, tergantung dari ukuran atau jumlah data. Namun, untuk dataset berukuran besar, proporsi pembagian 90:10 atau 99:1 juga umum dilakukan. Misal jika ukuran dataset sangat besar berisi lebih

dari 1 juta *record*, maka kita dapat mengambil sekitar 10 ribu data saja untuk testing alias sebesar 1% saja.

Pada modul ini, kita akan belajar membagi dataset dengan fungsi `train_test_split` dari library `sklearn`. Perhatikan contoh kode berikut.

```
1. from sklearn.model_selection import train_test_split  
2. x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1 )
```

Dengan fungsi `train_test_split` dari library `sklearn`, kita membagi array `X` dan `y` ke dalam 20% data testing (`test_size=0.2`). Misal total dataset `A` yang kita miliki adalah 1000 record, dengan `test_size=0.2`, maka data testing kita berjumlah 200 record dan jumlah data training sebesar 800 (80%).

Sebelum proses pemisahan, fungsi `train_test_split` telah mengacak dataset secara internal terlebih dahulu. Jika tidak, data testing hanya akan berisi semua data pada kelas tertentu saja. Misal dataset `A` kita terdiri dari 5 kelas dengan jumlah masing-masing kelas sebesar 200 record, maka dengan proses *shuffling* sebelum pemisahan, data testing akan memiliki data dari 5 kelas yang ada. Tanpa proses *shuffling*, seluruh data dari kelas 1 - 4 akan berakhir di set data training, dan data testing hanya berisi data dari kelas 5 saja. Proses *shuffling* menjaga rasio informasi pada data training dan testing tetap berimbang.

Melalui parameter `random_state`, fungsi `train_test_split` menyediakan *random seed* yang tetap untuk internal pseudo-random generator yang digunakan pada proses *shuffling*. Umumnya, nilai yang digunakan adalah 0, atau 1, atau ada juga yang menggunakan 42. 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, yang juga akan membuat akurasi model ML menjadi berbeda tiap kali di-*run*.

Berikut adalah contoh kode untuk memahami bagaimana penentuan `random_state` bekerja pada dataset.

```
1. from sklearn.model_selection import train_test_split  
2.  
3. X_data = range(10)  
4. y_data = range(10)
```

5.

```
6. print("random_state ditentukan")
```

```
7. for i in range(3):
```

```
8.     X_train, X_test, y_train, y_test = train_test_split(X_data, y_
    data, test_size = 0.3, random_state = 42)
```

```
9.     print(y_test)
```

10.

11.

```
12. print("random_state tidak ditentukan")
```

```
13. for i in range(3):
```

```
14.     X_train, X_test, y_train, y_test = train_test_split(X_data, y
    _data, test_size = 0.3, random_state = None)
```

```
15.     print(y_test)
```

Output:

1. random_state ditentukan

```
2. [3, 8, 4]
```

```
3. [3, 8, 4]
```

```
4. [3, 8, 4]
```

5.

6. random_state tidak ditentukan

```
7. [9, 2, 0]
```

```
8. [3, 8, 5]
```

```
9. [1, 4, 0]
```


1.13 Latihan SKLearn Train Test Split

Kode program yang akan diajarkan di sini bisa diunduh di tautan [berikut ini](#). Untuk membukanya, upload berkasnya Google Colab.

Tujuan

Pada codelab ini kita akan belajar membagi dataset menggunakan fungsi Train Test Split dari *library* SKLearn.

Tahapan Latihan

Untuk latihan membagi dataset terdiri dari tahapan-tahapan sebagai berikut:

1. Persiapkan dataset ke dalam Notebook.
2. Impor library SKLearn.
3. Buat variabel untuk menampung data training dan data testing.
4. Panggil fungsi `train_test_split()`.

Codelab

Pada Google Colab, import library yang dibutuhkan.

1. `import sklearn`
2. `from sklearn import datasets`

Library sklearn menyediakan **dataset iris** yakni sebuah dataset yang umum digunakan untuk masalah klasifikasi. Dataset ini memiliki jumlah 150 sampel. Untuk mendapatkan dataset, kita bisa menulis kode berikut pada cell baru.

1. `# load iris dataset`
2. `iris = datasets.load_iris()`

Dataset iris dari library sklearn belum dapat langsung dipakai oleh sebuah model ML. Sesuai dengan yang telah dibahas pada modul terdahulu, kita harus memisahkan antara atribut dan label pada dataset.

```
1. # pisahkan atribut dan label pada iris dataset
```

```
2. x=iris.data
```

```
3. y=iris.target
```

Untuk membuat train set dan test set kita tinggal memanggil fungsi `train_test_split`. `Train_test_split` memiliki parameter `x` yaitu atribut dari dataset, `y` yaitu target dari dataset, dan `test_size` yaitu persentase test set dari dataset utuh. `Train_test_split` mengembalikan 4 nilai yaitu, atribut dari train set, atribut dari test set, target dari train set, dan target dari test set.

```
1. from sklearn.model_selection import train_test_split
```

```
2.
```

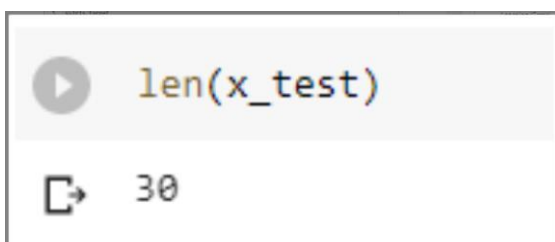
```
3. # membagi dataset menjadi training dan testing
```

```
4. x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
```

Ketika kita print panjang dari `x_test`, kita bisa melihat bahwa panjang dari atribut test set adalah **30 sampel**, sesuai dengan parameter yang kita masukkan pada fungsi `train_test_split` yaitu 0.2 atau 20% dari 150 sampel. Kode untuk print panjang dari `x_test` seperti di bawah ini.

```
1. # menghitung panjang/jumlah data pada x_test
```

```
2. len(x_test)
```



Pada tahap ini dataset kita telah siap dipakai untuk pelatihan model machine learning.

1.14 Data Evaluation

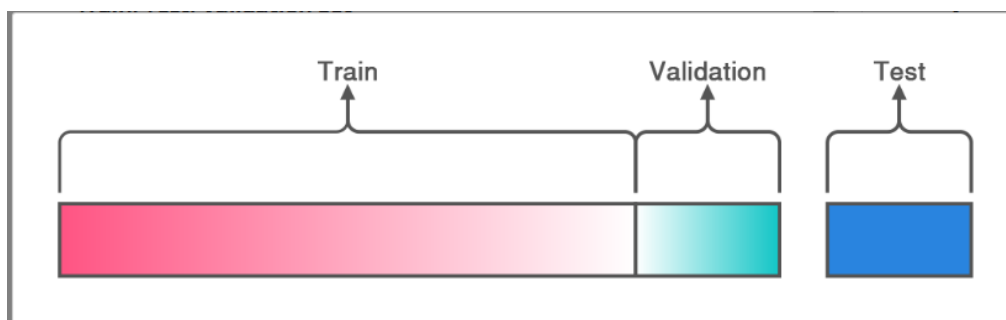
Pada submodul sebelumnya kita telah belajar bagaimana menggunakan test set untuk mengevaluasi model sebelum masuk ke tahap produksi.

Sekarang bayangkan ketika kita bertugas untuk mengembangkan sebuah proyek ML. Kita bimbang kala memilih model yang akan dipakai dari 10 jenis model yang tersedia. Salah satu opsinya adalah dengan melatih semua model tersebut lalu membandingkan tingkat erornya pada test set. Setelah membandingkan performa semua model, Anda mendapati model regresi linier memiliki tingkat eror yang paling kecil katakanlah sebesar 5%. Anda lalu membawa model tersebut ke tahap produksi.

Kemudian ketika model diuji pada tahap produksi, tingkat eror ternyata sebesar 15%. Kenapa ini terjadi? Masalah ini disebabkan karena kita mengukur tingkat eror berulang kali pada test set. Kita secara tidak sadar telah memilih model yang hanya bekerja dengan baik pada test set tersebut. Hal ini menyebabkan model tidak bekerja dengan baik ketika menemui data baru. Solusi paling umum dari masalah ini adalah dengan menambahkan *validation set* pada model machine learning.

Train, Test, Validation Set

Validation set atau *holdout validation* adalah bagian dari *train set* yang dipakai untuk pengujian model pada tahap awal. Secara sederhana, kita menguji beberapa model dengan *hyperparameter* yang berbeda pada data training yang telah dikurangi data untuk validation. Lalu kita pilih model serta hyperparameter yang bekerja paling baik pada validation set. Setelah proses pengujian pada holdout validation, kita bisa melatih model menggunakan data training yang utuh (data training termasuk data validation) untuk mendapatkan model final. Terakhir kita mengevaluasi model final pada test set untuk melihat tingkat erornya.

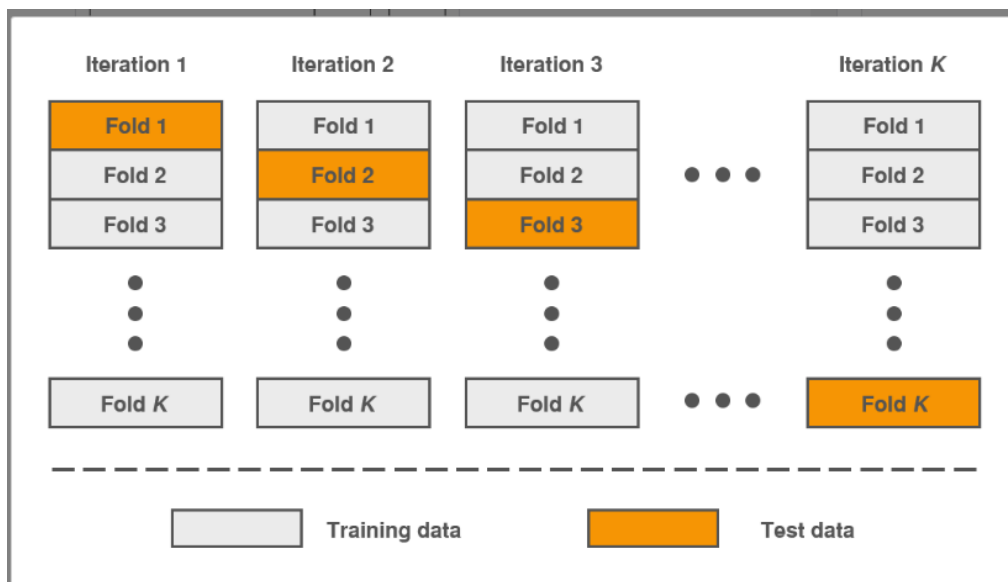


Dalam menggunakan holdout validation, ada beberapa hal yang harus dipertimbangkan. Jika ukuran *validation* set-nya terlalu kecil, maka ada kemungkinan kita memilih model yang tidak optimal. Sebaliknya, ketika ukurannya terlalu besar, maka sisa data pada train set lebih kecil dari data train set utuh. Kondisi ini tentu tidak ideal untuk membandingkan

model yang berbeda pada data training yang lebih kecil. Solusi untuk masalah ini adalah dengan menggunakan Cross Validation.

Cross Validation

K-Fold Cross Validation atau lebih sering disebut cross validation adalah salah satu teknik yang populer dipakai dalam evaluasi model ML. Pada cross validation dataset dibagi sebanyak K lipatan. Pada setiap iterasi setiap lipatan akan dipakai satu kali sebagai data uji dan lipatan sisanya dipakai sebagai data latih. Dengan menggunakan cross validation kita akan memperoleh hasil evaluasi yang lebih akurat karena model dievaluasi dengan seluruh data. Berikut adalah ilustrasi dari K-cross validation.



1.15 Latihan SKLearn Cross Validation Split

Kode program yang akan diajarkan di sini bisa diunduh di tautan [berikut ini](#). Untuk membukanya upload berkasnya ke Google Colab.

Tujuan

Pada codelab kali ini kita akan menggunakan *cross_validation_score* pada *classifier decision_tree*. Dataset yang digunakan adalah dataset iris.

Tahapan Latihan

Tahapan yang dilakukan pada codelab ini sebagai berikut:

1. Impor library yang dibutuhkan.
2. Pisahkan antara atribut dan label pada dataset.
3. Buat model *decision tree*.
4. Hitung hasil *cross validation* dari model dengan fungsi *cross_val_score()*.

Codelab

Para pengembang ML biasanya meng-*import* semua *library* yang dibutuhkan di *cell* pertama. Dalam tahap latihan ini, Anda akan melakukan *import library* pada *cell* yang berkaitan saja. Tujuannya adalah agar Anda memahami fungsi setiap *library* yang digunakan dalam model ML yang Anda buat.

Dataset yang akan kita gunakan adalah dataset iris yang dipakai pada submodul sebelumnya.

1. `import sklearn`
2. `from sklearn import datasets`
- 3.
4. `# Load iris dataset`
5. `iris = datasets.load_iris()`

Kemudian kita bagi antara atribut dan label pada dataset.

```
1. # mendefinisikan atribut dan label pada dataset
```

```
2. x=iris.data
```

```
3. y=iris.target
```

Kita akan membuat model machine learning pertama kita yaitu decision tree, menggunakan *library scikit learn*. Model machine learning juga sering disebut sebagai classifier. Lebih lanjut, variabel `clf` adalah singkatan dari *classifier*.

```
1. from sklearn import tree
```

```
2.
```

```
3. # membuat model dengan decision tree classifier
```

```
4. clf = tree.DecisionTreeClassifier()
```

Setelah dataset dan model siap, kita bisa menggunakan cross validation untuk mengevaluasi performa dari model machine learning. Fungsi `cross_val_score()` seperti di bawah menerima 4 parameter yaitu, 'clf' yang merupakan model machine learning, 'X' yang merupakan atribut dari dataset, 'y' yang merupakan label dari dataset, dan 'cv' yang merupakan jumlah *fold* yang akan dipakai pada *cross validation*.

```
1. from sklearn.model_selection import cross_val_score
```

```
2.
```

```
3. # mengevaluasi performa model dengan cross_val_score
```

```
4. scores = cross_val_score(clf, x, y, cv=5)
```

`Cross_val_score` mengembalikan nilai berupa larik atau array yang terdiri dari akurasi pengujian setiap *fold* dari dataset. Untuk mencetak dan mengetahui hasilnya, tambahkan kode `scores` di bawah kode sebelumnya. Tampilannya seperti gambar di bawah ini.

```
scores  
array([0.96666667, 0.96666667, 0.9        , 0.93333333, 1.        ])
```

Elemen pertama dari larik menunjukkan nilai 0.96666 yang berarti ketika *fold* pertama dijadikan *validation set* dan fold lainnya dijadikan train set, hasil dari pengujian tersebut adalah akurasi sebesar 0.96666.

Melihat akurasi dari seluruh pengujian fold yang memiliki nilai tinggi dan konsisten pada tiap fold, kita mendapatkan gambaran bahwa model kita memiliki performa yang sangat baik.

Secara umum jika hasil dari pengujian tiap fold pada *cross validation* memiliki nilai yang bervariasi dari 0.85 sampai 0.99, maka model tersebut dapat dikatakan baik.