MACHINE LEARNING

Teori, Studi Kasus dan Implementasi Menggunakan Python

Ibnu Daqiqil Id

MACHINE LEARNING

Teori, Studi Kasus dan Implementasi
Menggunakan Python

Edisi 1

Ibnu Daqiqil ID

Judul : MACHINE LEARNING: Teori, Studi Kasus dan Implementasi Menggunakan Python
Penulis : Ibnu Daqiqil Id
ISBN : 978-632-255-092-6

Diterbitkan Oleh **UR PRESS**, Juni 2021

Alamat Penerbit
Badan Penerbit Universitas Riau
UR PRESS Jl. Pattimura No. 9, Gobah Pekanbaru 28132,
Riau, Indonesia
Telp. (0761) 22961, Fax. (0761) 857397
e-mail: unri_press@yahoo.co.id
ANGGOTA IKAPI

Lisensi

Prakata

" Ilmu itu bagaikan hewan buruan dan menulis itu pengikatnya **…"**
-Imam Syafi'i

Machine Learning (ML) sudah menjadi bagian dalam kehidupan sehari-hari. Dengan mengimplementasikan ML saat ini kita dapat menikmati kemudahan-kemudahan seperti memerintahkan *Smartphone* untuk memutarkan lagu favorit. Selain itu, kita dapat mengimplementasikannya pada banyak jenis kasus mulai dari diagnosa perawatan medis hingga mendapatkan rekomendasi film yang harus ditonton.

Penulis mencoba mengemas buku ini secara praktis, tidak berbelit-belit dan langsung tepat pada sasaran dan disertai dengan contoh implementasi menggunakan Bahasa Python. Buku ini cocok untuk dibaca oleh pemula yang baru mulai belajar tentang *Machine Learning*. Penulis mencoba menyajikan teori-teori dasar secara ringkas, sehingga pembaca akan diarahkan memahami lebih dalam tentang ML dengan memberikan ilustrasi dan analogi yang mudah dipahami.

Buku disusun atas empat kategori diantaranya Pengenalan ML; Teori Pendukung ML; ML *Lifecyle* dan Algoritma-Algoritma ML. Bagi pembaca yang baru mempelajari, disarankan agar membaca buku secara berurutan. Namun apabila telah memiliki pengetahuan awal tentang ML dapat memulai dari ML *Lifecyle*. Selanjutnya pada bagian algoritma-algoritma ML, penulis memilih algoritma-algoritma dasar yang dibagi menjadi dua kelompok yaitu *Supervised Learning* dan *Unsupervised Learning*. Segala materi, dataset dan Source Code pada buku ini dapat diakses di http://ibnu.daqiqil.id

Dengan segala kerendahan hati dan keterbukaan, penulis menyampaikan rasa terima kasih kepada pihak-pihak yang telah

membantu proses pembuatan buku ini terutama LPPM Universitas Riau serta kritik agar buku ini dapat menuju kesempurnaan. Akhir kata, penulis berharap agar buku ini dapat membawa manfaat kepada pembaca. Secara khusus, penulis berharap semoga buku ini dapat menginspirasi generasi bangsa ini agar menjadi generasi yang tanggap dan tangguh.

Okayama, Maret 2021

Penulis

Daftar Isi

Daftar Gambar

To My parent, My Family and My Teacher

Bab

1

Perkenalan *Machine Learning*

Sesungguhnya bersama kesulitan itu ada kemudahan. (Q.S Al-Insyirah: 6)

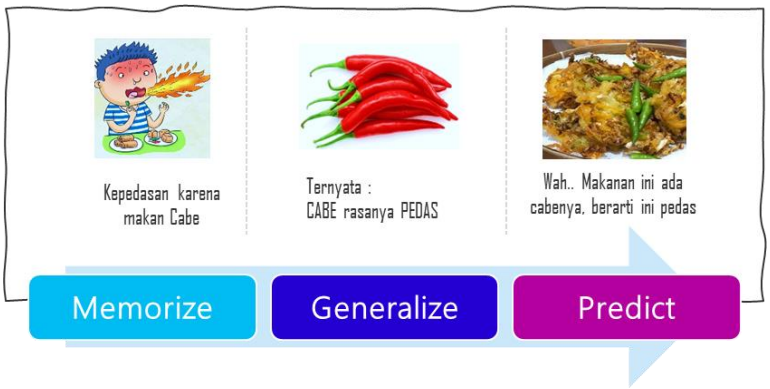
Topik Pembelajaran

1. Intuisi dan definisi *Machine Learning*
2. Bagaimana cara proses manusia dan mesin belajar
3. Apa aplikasi *Machine Learning* di kehidupan sehari-hari
4. Kenapa harus belajar *Machine Learning*?

Manusia adalah makhluk pembelajar yang memiliki kemampuan belajar yang sangat luar biasa. Proses pembelajaran tersebut telah dimulai sejak dia dilahirkan sampai akhir hayatnya. Seiring bertambah usia, manusia mulai mempelajari bagaimana cara berjalan, berbicara, berkomunikasi, membaca, menulis dan berpikir untuk membuat mengolah informasi, lalu bertindak dan membuat keputusan. Salah satu cara manusia belajar dan mengambil keputusan adalah berdasarkan pengalaman. Sebagai contoh ketika seorang anak menemukan cabai untuk pertama kalinya, maka tanpa ragu anak tersebut akan melahap cabai tersebut. Akibatnya dia akan merasa "kepedasan" dan "tidak nyaman". Pengalaman kepedasan tersebut karena memakan cabai akan menjadi sebuah pembelajaran baginya. Dari pengalaman tersebut sang anak belajar bahwa cabai memiliki sifat pedas dan dapat menimbulkan rasa tidak nyaman, maka dia tidak akan berani untuk melahap cabai tersebut atau setidaknya akan berhati-hati terhadap cabai.

Berdasarkan contoh tersebut, dapat disimpulkan bahwa salah satu proses pengambilan keputusan adalah berdasarkan pengalaman sebelumnya. Proses pengambilan keputusan berdasarkan pengalaman setidaknya melibatkan tiga proses berikut (Gambar 1.1):

1. ***Memorize***. Anak tersebut mengingat semua data dan kejadian yang telah terjadi ataupun kejadian yang mirip dan berkaitan. Bagaimana rasa cabai, bentuk cabai, atau warna cabai.
2. ***Generalize***. Berdasarkan data-data tersebut, si anak akan memformulasikan sebuah "pola" atau "*general rule*" pada sebuah kejadian tadi.
3. ***Predict***. Ketika menghadapi kasus yang berkaitan atau mirip, maka rule atau pola tersebut diaktifkan untuk memprediksi apa tindakan yang akan dilakukan. Misalnya si anak di berikan makanan yang berisi potongan cabai, maka dengan cepat dia akan menunjukkan responsnya.



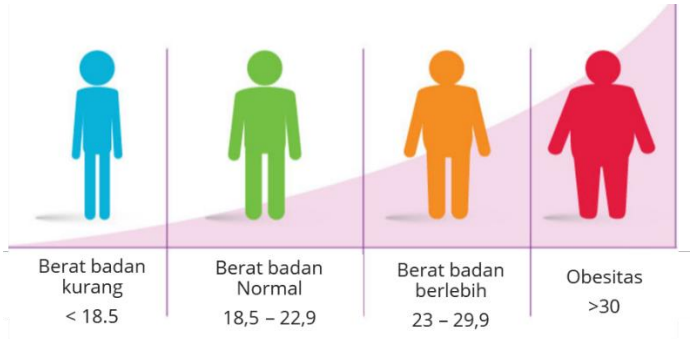
Gambar 1.1. Kerangka mengambil keputusan berdasarkan pengalaman

Komputer berbeda dengan manusia, ia hanya mengikuti instruksi yang telah ditulis oleh programmer-nya. Agar dapat menyelesaikan sebuah masalah, komputer membutuhkan sebuah algoritma yang berisi langkah-langkah instruksi yang harus dikerjakan komputer untuk memecahkan sebuah masalah atau untuk mengubah *input* menjadi *output*. Instruksi-

instruksi tersebut direalisasikan dalam bahasa pemrograman tertentu, dan tugas komputer hanya mengeksekusi baris-baris perintah tersebut.

Sebagai contoh perhatikan Gambar 1.2 yang berisi informasi kategori kondisi berat badan berdasarkan BMI (*body mass index*) atau Indeks Massa Tubuh. Kategori tersebut dapat dibagi menjadi 4 kelompok yaitu *Underwight, Normal, Overwight* dan *Obese*. Indeks Massa Tubuh adalah indeks yang paling mudah dan paling sering digunakan untuk menggolongkan tingkat kegemukan dan obesitas. Cara perhitungannya adalah dengan membagi berat badan seseorang dalam kilogram dengan kuadrat dari tinggi badannya dalam meter (kg/m2). Setelah mendapatkan nilai BMI maka aturan berikut digunakan untuk menentukan kelompok:

- Jika nilai BMI lebih kecil dari 18.5 maka Anda dikategorikan *Underwight*
- Jika BMI di antara 18.5 sampai 24.9 maka dikategorikan *Normal*
- Jika nilai BMI di antara 25 sampai 29.9 maka dikategorikan *Overwight*
- Jika besar dari 30 dikategorikan *Obese*.



Gambar 1.2. Kategori Berat Badan berdasarkan BMI

Implementasi dari pengukuran BMI ke dalam aplikasi komputer sangat sederhana. Dengan menggunakan seleksi (perintah *if/else*) maka permasalahan akan terselesaikan. Aplikasi yang dihasilkan akan dapat mengategorikan dengan akurat. Jika dianalisis, proses kategorisasi yang dilakukan berawal dari sebuah standar yang baku, di mana komputer hanya diberikan sejumlah instruksi lalu mengerjakan sesuai instruksi tersebut.

Pada kasus ini aplikasi tidak memiliki kemampuan belajar dan hanya mengikuti instruksi yang telah ditentukan.

Berbeda dengan Standar BMI, dalam perspektif *Machine Learning*, sebuah komputer berusaha belajar dengan data yang diberikan. Komputer **hanya** diberi sejumlah data dari 4 kategori BMI lalu mempelajarinya. Misalnya, komputer diberikan daftar berat badan dari 20 orang yang berasal dari 4 kategori tersebut di mana:

- Kategori *Underwight* adalah {17, 18,16,18,16.5}
- Kategori *Normal* adalah {19, 20, 22, 21, 23}
- Kategori *Overwight* adalah {29, 26, 27, 28, 25}
- Kategori *Obese* adalah {49, 50, 32, 41, 33}.

Berdasarkan data tersebut komputer mempelajari bagaimana karakteristik orang per kategorinya sehingga mampu memprediksi kategori BMI ketika diberi masukan berat badan.

Kenapa tidak memprogram semuanya? Ada beberapa kasus di mana tidak ada algoritma yang cocok untuk menyelesaikan kasus tersebut, contohnya pendeteksian email spam. Kasus ini berbeda dengan kasus BMI, email yang berisi spam tidak memiliki aturan khusus (kriteria yang baku sebuah dokumen dikatakan spam) dan cenderung berbeda antara satu dengan yang lain dan dapat berubah-ubah seiring waktu. Kasus spam hanya dapat diselesaikan jika kita memiliki **pengetahuan** tentang isi atau ciri dokumen spam. Isi dan ciri-ciri itu sangat berbeda antara satu pengguna dengan yang lainnya, tergantung preferensi masing-masing.

Sebagai contoh seorang *marketing* maka sebagian besar emailnya berisi promosi-promosi produk, diskon, penawaran, *invoice* dan lain-lain. Menariknya, sebagian email tersebut, misalnya promosi produk tersebut dapat dianggap spam bagi seorang pelajar yang belum memiliki penghasilan. Jadi semuanya tergantung dari preferensi pengguna. Pada kasus ini kita tidak memiliki aturan yang baku mengenai dokumen spam. Ada banyak kasus-kasus seperti kasus spam, sehingga ahli komputer

mencoba mereplika proses pembelajaran manusia ke komputer dengan tujuan komputer dapat menyelesaikan permasalahan-permasalahan seperti ini dengan lebih efektif maka muncullah istilah *Machine Learning*.

Contoh lainnya kenapa pemrograman secara ekplisit tidak efektif adalah pendeteksian digit (0-9) pada gambar tulisan tangan. Pada gambar yang berukuran 100x100 pixel, jumlah kombinasi dari *input* dari satu digit aja akan sangat beragam sehingga tidak mungkin untuk membuat sebuah program yang ditulis secara explisit. Gambar 1.7 memperlihatkan sebagian gambar digit tulisan tangan. Untuk mengenali angka 0, akan ada banyak sekali kemungkinan untuk membuat huruf tersebut.

Gambar 1.3. Digit angka yang ditulis menggunakan tulisan tangan (sumber: lecun.com)

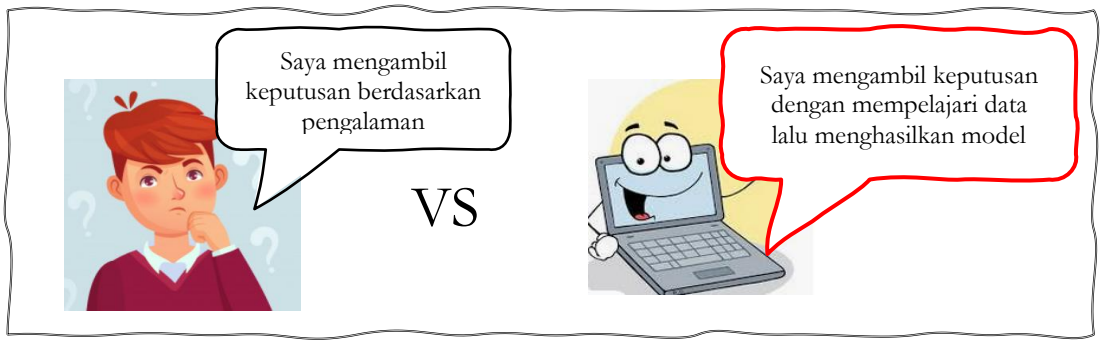
1.1 Definisi Machine Learning

Machine Learning (ML) merupakan bidang studi yang fokus kepada desain dan analisis algoritma sehingga memungkinkan komputer untuk dapat belajar. Menurut Samuel, ML berisi sebuah algoritma yang bersifat *generic* (umum) dimana algoritma tersebut dapat menghasilkan sesuatu yang menarik atau bermanfaat dari sejumlah data tanpa harus menulis kode yang spesifik. Pada intinya, algoritma yang generik tersebut ketika diberikan sejumlah data maka ia dapat membangun sebuah aturan atau model atau inferensi dari data tersebut. Sebagai contoh sebuah Algoritma untuk mengenali tulisan tangan dapat digunakan untuk mendeteksi email yang berisi spam dan bukan spam tanpa mengganti kode. Algoritma yang sama ketika diberikan data pelatihan yang berbeda menghasilkan logika klasifikasi yang berbeda.

DEFINISI

Machine Learning adalah salah satu bidang cabang ilmu komputer yang memberikan kemampuan kepada komputer untuk dapat belajar tanpa diprogram secara explisit (Arthur Samuel, 1959)

Machine learning juga dapat diartikan sebuah komputer yang memiliki kemampuan belajar tanpa diprogram secara eksplisit. Program tersebut memanfaatkan data untuk membangun model dan mengambil keputusan berdasarkan model yang telah dibangun.



Gambar 1.4. Manusia vs mesin dalam mengambil keputusan

Selain Samuel, Mitchel(1997) juga memberikan sebuah definisi ringkas dan jelas mengenai ML dimana *Machine Learning* adalah "Satu program komputer yang dikatakan telah melakukan pembelajaran dari pengalaman **E (*Experience*)** terhadap tugas **T (*Task*)** dan mengukur peningkatan kinerja **P (*Performance Measure*),** jika kinerja Tugas T diukur oleh kinerja P, maka meningkatkan pengalaman E". Dari definisi ini Mitchel dapat dikatakan sebuah aplikasi *Machine Learning* memiliki 3 komponen yaitu Task T, *Performance Measure* P, dan *Experience* E. Oleh karena itu, untuk membangun sebuah aplikasi ML maka komponen T, P dan E harus dapat di identifikasi.

Task T merupakan objektif dari program ML yang kita buat. Program ML membantu kita untuk mengatasi masalah yang sulit untuk diselesaikan oleh program tradisional yang bersifat statis. Berikut ini beberapa contoh *Task* yang ada program ML:

1. **Klasifikasi** adalah suatu pengelompokan data di mana data yang digunakan tersebut mempunyai kelas label atau target. Pada jenis *task* ini, program komputer diminta untuk "menebak" atau memilih kelompok/kategori/kelas dari data. Pada task ini kelas telah ditetapkan sebelumnya. Contohnya adalah klasifikasi emosi pada foto manusia dengan tiga kelas yaitu *Happy, Angry* dan *Neutral.* Aplikasi tersebut harus memilih salah satu (dalam kasus tertentu bisa lebih dari satu) dari tiga kelas tersebut berdasarkan foto yang diberikan.
2. **Regresi** sederhananya adalah sebuah tugas di mana aplikasi ML diminta untuk menebak angka (bilangan continue) berdasarkan sejumlah data. Sebagai contoh kita ingin melakukan prediksi harga saham, maka luaran dari prediksi tersebut adalah sebuah nilai harga. Jadi salah satu perbedaan antara regresi dan klasifikasi adalah objek yang diprediksi, regresi memprediksi angka sedangkan klasifikasi memprediksi kelas.
3. **Transkripsi** adalah sebuah tipe *Task* dimana aplikasi ML mencoba "memahami" sebuah data yang tidak terstruktur yang merepresentasikan data lain. Sebagai contoh pengenalan tulisan pada

gambar dimana gambar berisi data pixel yang tidak terstruktur, tetapi data tersebut merepresentasikan sebuah tulisaan atau huruf-huruf. Contoh lainnya adalah aplikasi *text to speech* dimana data berupa gelombang berisi tekt. Proses mengubah gelombang suara menjadi teks adalah transkripsi.

4. ***Machine Translation.*** *Task* ini merupakan proses mengubah sekumpulan *input* terurut yang berisi simbol-simbol tertentu menjadi simbol tertentu dibahasa lainnya. Contoh aplikasinya penerjemah bahasa Indonesia dan bahasa Inggris.
5. ***Anomaly Detection.*** *Task* ini bertujuan untuk menggunakan aplikasi ML untuk memeriksa sejumlah data atau event lalu menandainya sebagai sesuatu yang tidak biasa.
6. ***Syntesis* dan *Sampling.*** *Task* ini bertujuan untuk membuat atau *generate* sesuatu berdasarkan contoh. Misalnya aplikasi ML yang dapat menghasilkan sebuah gambar pemandangan berdasarkan sketsa tertentu.

Untuk mengevaluasi kinerja aplikasi ML maka diperlukan sebuah metode pengukuran kinerja yang kualitatif. Setiap *task* memiliki metode pengukuran yang berbeda sebagai contoh antara klasifikasi dan regresi cara mengukur performanya berbeda. Klasifikasi mengukur kinerja berdasarkan perbandingan jumlah tebakan benar dan salah sedangkan regresi dinilai berdasarkan kedekatannya dengan nilai asli dari suatu tebakan. Jenis-jenis metode evaluasi dapat dilihat pada bab 3.

Aplikasi atau model ML mendapatkan *experience* berdasarkan *dataset* yang disediakan pada proses pelatihan. Sebuah *dataset* adalah kumpulan contoh-contoh yang harus dipelajari oleh komputer agar dapat menyelesaikan *Task*-nya. Informasi detail mengenai *dataset* akan dibahas pada bab berikutnya.

1.2 Sejarah Singkat ML

Arthur Samuel, seorang pionir dalam pengembangan permainan komputer dan kecerdasan buatanlah yang pertama kali mengeluarkan istilah "*Machine Learning*" ke publik pada tahun 1959. Perkembangan pembelajaran mesin tumbuh berkat berkembangnya bidang kecerdasan buatan atau *Artificial Intelegence* (AI). Banyak peneliti di bidang AI tertarik untuk memiliki mesin yang dapat belajar dari data. Para peneliti berusaha untuk mendekati masalah dengan berbagai metode simbolik, serta apa yang kemudian disebut Neural Network, Penalaran probabilistik dan berbagai model statistik.

Era sebelum-1920an

Thomas Bayes, Adrien-Marie Legendre, Andrey Markov, dan matematikawan lainnya memulai penelitian yang mejadi teknik dasar pada konsep-konsep ML. Sejarah Machine learning dapat dikatakan dimulai oleh penelitian Thomas Bayes pada tahun 1763 yang dipublikasikan oleh temannya Richard Prince. Penelitian tersebut disempunakan oleh Pierre-Simon Laplace sehingga sekarang dikenal dengan Theorema Bayes (1812). Selanjutnya penelitian penting lainnya dilakukan oleh Adrien-Marie Legendre matematikawan asal Prancis, dimana dia mengembangkan metode Least Squares yang bertujuan untuk melakukan *data fitting* menggunakan pendekatan aljabar (1805), serta Andrey Markov mendeskripsikan sebuah teknik yang ia gunakan untuk menganalisis puisi. Teknik tersebut dikenal dengan Markov Chains (1913). Teori-teori mereka tersebut menjadi landasan utama untuk machine learning.

Era 1940-1950an

Pada tahun 1943, McMulloh dan Pitts mengembangkan model matematis yang terinspirasi dari bagaimana otak manusia bekerja, dimana dalam model tersebut terdapat neuron-neuron yang dapat menjadi aktif atau nonaktif seperti saklar *on-off*. Selain itu neuron-neuron tersebut memiliki kemampuan

untuk belajar dan menghasilkan respons yang berbeda berdasarkan input yang diberikan.

Sumbangan besar lainnya diberikan oleh Alan Turing, pada tahun 1950 yang menciptakan mesin turing untuk menjawab pertanyaan "Dapatkan Komputer Berfikir?". Paper Alan Turing pada tahun 1950 berjudul "*Computing Machinery and Intelligence*" mendiskusikan syarat sebuah mesin dianggap cerdas. Dia beranggapan bahwa jika mesin dapat dengan sukses berprilaku seperti manusia, kita dapat menganggapnya cerdas.

Pengembangan selanjutnya, pada tahun 1959, Allen Newell dan Herbert Simon mengembangkan sebuah proyek disebut *General Problem Solver* (GPS). GPS telah berhasil menyelesaikan permasalahan manusia menggunakan teknik *mean-ends analysis*.

Era 1960-1990an

Selanjutnya penemuan fenomenal lainnya pada tahun 1962, dimana Frank Rosenblatt membuktikan teorema Perceptron dan berhasil mengklasifikasi objek dari dua kelompok menggunakan *classification rules*.

Perkembangan selanjutnya dibuat oleh Rumelhart (1986) dengan mencoba mengembangkan sistem layar tunggal (*single layer*) pada Perceptron menjadi sistem layar jamak (*multilayers*), yang kemudian disebut dengan sistem *back-propagation*. Konsep *back-propagation* telah ada sebelumnya, namun Rumelhart dan Hinton-lah yang pertama kali mempopulerkan dan membuktikan metode ini dapat dilakukan. Setelah itu, muncul beberapa model jaringan saraf tiruan lain yang dikembangkan oleh Kohonen (1972), Hopfield (1982), dan lain-lain.

Selain algoritma berbasis Neural Network, di Rusia berkembang algoritma-algoritma *Generalized Portrait* pada tahun 1960-an (1963, 1964). Algoritma tersebut berakar dari teori pembelajar statistika yang telah dikembangkan selama 3 dekade oleh Vapnik dan Chervonenkis. SVM (*Support Vector Machines*) diperkenalkan oleh Vapnik, Boser, dan Guyer

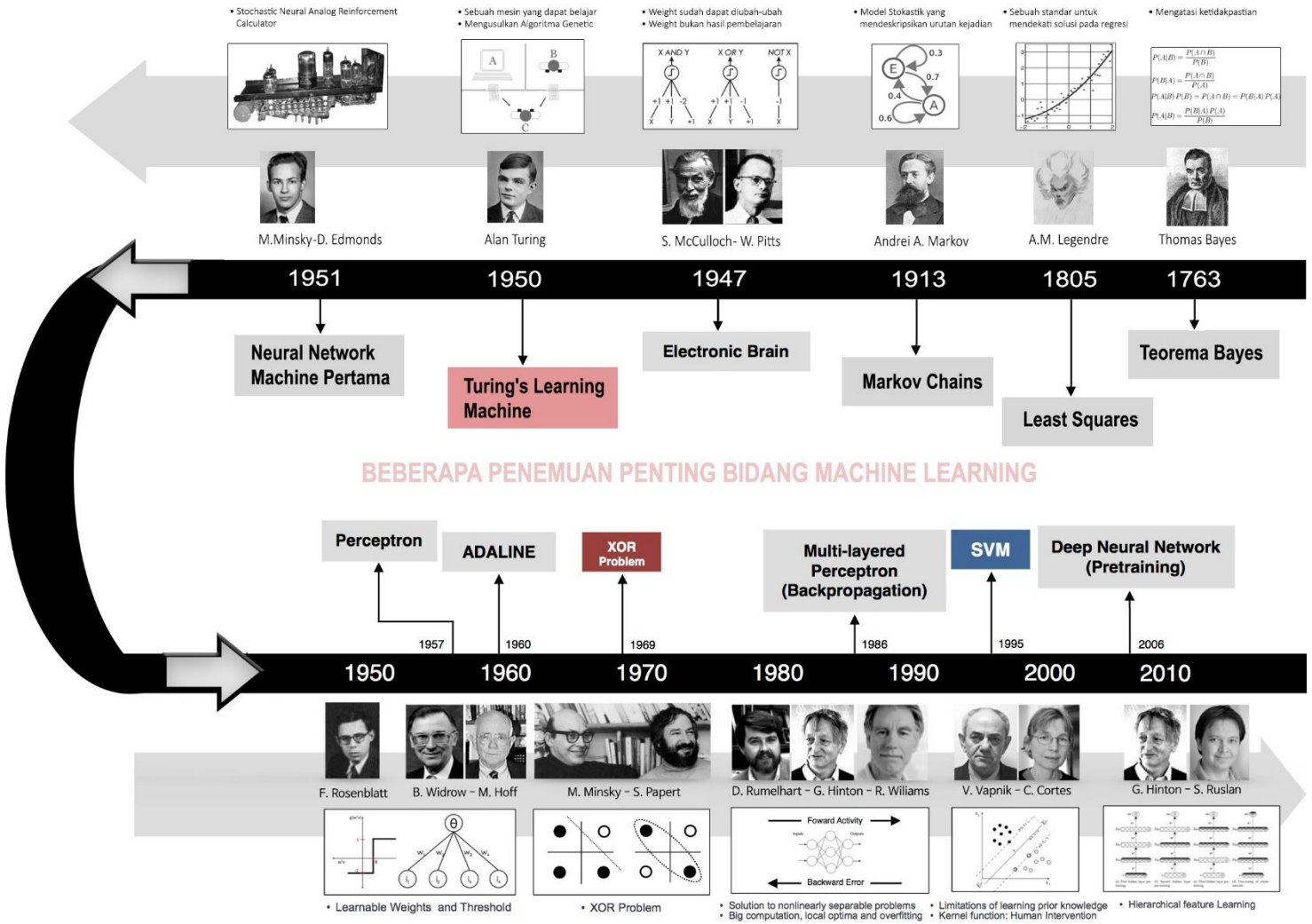
dalam konferensi COLT tahun 1992 dalam sebuah jurnal dan berkembang dengan sangat cepat.

SVM pertama kali diperkenalkan ke dunia sejak akhir tahun 1970-an (Vapnik,1979), *Support Vector Machines* tidak mendapat banyak perhatian oleh para peneliti. Kemudian Vladimir Vapnik kembali menerbitkan bukunya pada tahun 1990-an (Vapnik, 1995; Vapnik, 1998). Sejak saat itu, SVM menjadi metode yang popular dan aktif diteliti pada bidang Machine Learning.

Era 2000an

Di akhir 90-an hingga pertengahan tahun 2000-an, neural networks sempat "nyaris dilupakan" dikarenakan muncul berbagai algoritma seperti *Support Vector Machines*, *AdaBoost* yang dapat dieksekusi lebih cepat dengan performa yang lebih baik pada waktu itu. Neural networks kembali mendapatkan perhatian ketika *Deep Belief Networks* (DBN) membuat terobosan dengan menjadi model *handwritten digit recognition* yang paling akurat, yang pada akhirnya memunculkan istilah Deep Learning.

Istilah *Deep Learning* makin popular dengan diperkenalkannya *Convolution Neural Network* (CNN). CNN dengan arsitektur tertentu yang dipadukan dengan berbagai trik (misalnya, *drop out regularization*, pemanfaatan Rectified Linear Unit (ReLU) sebagai fungsi aktivasi, data augmentation) sehingga telah mampu melakukan klasifikasi pada data gambar yang berjumlah sangat besar (ImageNet). ImageNet yang memiliki 1000 kategori objek dan dengan jumlah 1 juta gambar, hal ini melebihi performa manusia.



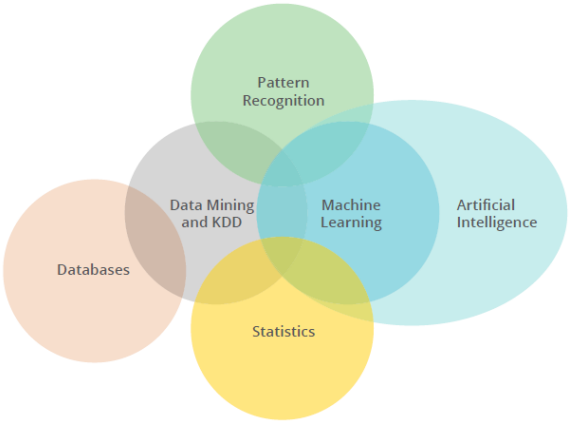
Gambar 1.5. Beberapa Penemuan penting dibidang Machine Learning

1.3 Machine Learning Sebagai Cabang AI

Jika kita tinjau kembali, bagaimana cara manusia untuk mengambil keputusan bukan hanya berdasarkan pengalaman tetapi juga dapat berdasarkan logika. Seperti yang telah dijelaskan, *Machine learning* adalah bagian dari *artificial intelligence* yang berfungsi untuk membuat komputer memiliki kemampuan untuk belajar tentang data baru tanpa harus diprogram secara ekplisit. Fokus utamanya adalah membangun sebuah aplikasi komputer yang dapat mempelajari data, lalu membuat sebuah model yang siap digunakan untuk memecahkan kasus tertentu.

Berbeda dengan *Data Mining*, dimana prosesnya dimulai dari pengumpulan *database* yang berukuran besar yang sebelumnya tidak dapat dipahami dan tidak diketahui dan kemudian menggunakan informasi itu untuk membuat keputusan bisnis yang relevan. Sederhananya, *Data Mining* dapat diartikan sejumlah metode yang digunakan untuk menemukan pengetahuan berupa relasi dan pola yang berasal dari *database* yang besar yang sebelumnya tidak diketahui.

Oleh karena itu, dapat dikatakan bahwa *Data Mining* sebagai titik temu berbagai bidang lain seperti kecerdasan buatan, basis data, pengenalan pola, visualisasi data, pembelajaran mesin, statistik, dan sebagainya. **Tujuan utama dari proses *Data Mining* adalah untuk mengekstraksi informasi dari berbagai set data dalam jumlah yang besar untuk mengubahnya dalam struktur yang tepat dan dapat dimengerti untuk penggunaan akhir.** Dapat disimpulkan *Data Mining* merupakan proses yang digunakan oleh para ilmuwan data dan penggemar pembelajaran mesin untuk mengubah sebuah *dataset* yang besar menjadi sesuatu yang lebih bermanfaat.



Gambar 1.6. Pembagian cabang-cabang ilmu data (Sumber: Towardsai.com)

Secara umum, ML dan *Data Mining* mengikuti proses yang relatif sama. Perbedaan utamanya adalah *Data Mining* didesain untuk mengekstak informasi dari sejumlah data yang besar, sedangkan *Machine Learning* bertujuan untuk membuat komputer mempelajari data. Sederhananya *Data Mining* adalah sebuah metode untuk mengekstrak luaran tertentu dari sejumlah *dataset* yang besar. Luarannya adalah sebuah informasi, sedangkan *Machine learning* luarannya adalah sebuah model yang dapat melakukan fungsi tertentu.

Pembelajaran mesin mengikuti metode analisis data yang bertanggung jawab untuk mengotomatisasi pembuatan model secara analitis. Menggunakan algoritma yang secara iteratif untuk mendapatkan pengetahuan dari data dan dalam proses ini memungkinkan komputer menemukan pengetahuan yang tampaknya tersembunyi tanpa bantuan dari eksternal.

Selain istilah *Data Mining*, ada sebuah istilah lagi yang cukup membingungkan yaitu *Data Science*. *Data science* berisi kegiatan-kegiatan seperti pengumpulan data terstruktur dan tidak terstruktur yang mencakup segala sesuatu yang terkait dengan pembersihan (*cleansing*), persiapan (*preparation*) dan analisis akhir data, lalu mengaplikasikan metode machine learning, analisa predictive untuk mengekstrak informasi penting dari

dataset. Informasi yang didapat itu selalu diambil dari sudut pandang bisnis sehingga dapat digunakan untuk prediksi ataupun *insight* yang dapat digunakan untuk mengambil keputusan bisnis. *Data science* menggabungkan pemrograman, penalaran logis, matematika dan statistik.

Sederhanya *Data Science* mencoba menangkap data dengan cara yang cerdik dan mendorong kemampuan melihat sesuatu dengan perspektif yang berbeda dikaitkan dengan kebutuhan dan strategi bisnis suatu perusahaan. Kekuatan *Data Science* adalah visualisasinya. Kemampuannya untuk dibidang visualisasi membuat cabang ilmu ini berbeda dengan *Machine Learning*, dimana dengan visualisasi yang tepat membuat pengguna mampu memanfaatkan pengetahuan ke dalam domain bisnis.

1.4 Contoh sederhana aplikasi ML

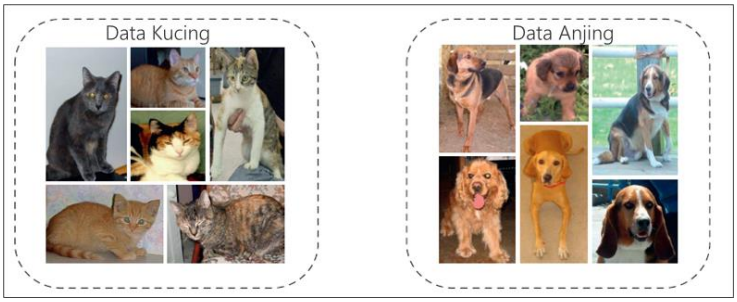
Membedakan anjing dan kucing bukanlah hal yang sulit bagi manusia, dengan memperhatikan beberapa contoh maka otak manusia telah dapat membedakan mana anjing dan kucing. Tidak perlu pendekatan saintis untuk membedakan mereka, cukup dengan mengidentifikasi karakter fisik, manusia telah dapat membedakan mereka. Pada bab ini kita akan mencoba mengajari bagaimana komputer dapat mengenali anjing dan kucing. Oleh karena itu perlu dilakukan beberapa tahap diantaranya:

> Penjelasan teknis secara detail tentang teknik yang digunakan pada subbab ini dapat dilihat pada Bab 3.
> Pada tahap ini diharapkan pembaca dapat memahami **intuisi** dari ML

1. Pengumpulan data gambar kucing dan anjing.

Seperti manusia, komputer harus diperkenalkan kepada objek yang akan dipelajari yaitu objek anjing dan objek kucing. Salah satu cara untuk "mengenalkan" kucing dan anjing adalah dengan cara memberikan sekumpulan data gambari anjing dan kucing. Pada kasus ini ANJING dan KUCING dapat dikatakan kategori atau kelas.

Tugas utama komputer adalah memasukkan sebuah data ke dalam salah satu kategori ini. Proses ini biasa disebut klasifikasi. Data yang digunakan untuk pembelajaran biasanya disebut dengan *data training. Data training* harus memiliki informasi yang dilengkapi dengan data kategori/label. Pada kasus ini kita akan menggunakan sekumpulan gambar yang telah diidentifikasi apakah gambar tersebut kucing atau anjing. Gambar 1.7 menunjukkan sebuah *dataset* yang telah diberi label Kucing dan Anjing. Data-data tersebut dapat dikumpulkan dari berbagai sumber atau dibuat sendiri oleh pengembang.



Gambar 1.7. Sekumpulan data kucing (kotak kiri) dan data anjing (kotak kanan) yang akan digunakan untuk mempelajari objek kucing dan anjing.

2. Mendesain Fitur kucing dan anjing.

Perhatikan, lalu pelajari apa ciri pembeda antara dua makhluk tersebut? Apakah warna, jenis bulu, ukuran atau jenis telinga, atau ukuran hidung? Sebagai contoh fitur warna, apakah kita dapat membedakan yang mana kucing atau anjing berdasarkan warna? Jika dapat maka fitur warna dapat digunakan, jika tidak coba cek fitur lainnya misalnya jenis telinga. Jika diperhatikan, ada perbedaan signifikan antara kuping anjing dan kucing, maka fitur ini dapat digunakan. Pada intinya temukan ciri/fitur yang dapat membedakan kedua binatang tersebut. Fitur-fitur yang telah ditemukan akan digunakan oleh komputer sebagai *data training*.

Proses menentukan fitur (ekstraksi fitur) sangat penting karena akan berpengaruh terhadap hasil yang ingin dicapai dan sangat spesifik terhadap kasus tertentu. Contohnya fitur "**jumlah kaki**", fitur ini tidak efektif digunakan pada kasus ini tetapi untuk membedakan kucing dan ayam maka fitur ini sangat penting. Selanjutnya fitur akan berbeda antara satu kasus dengan yang lain dan proses untuk menemukannya cukup menantang serta bisa menyulitkan karena kemiripan yang sangat detail.

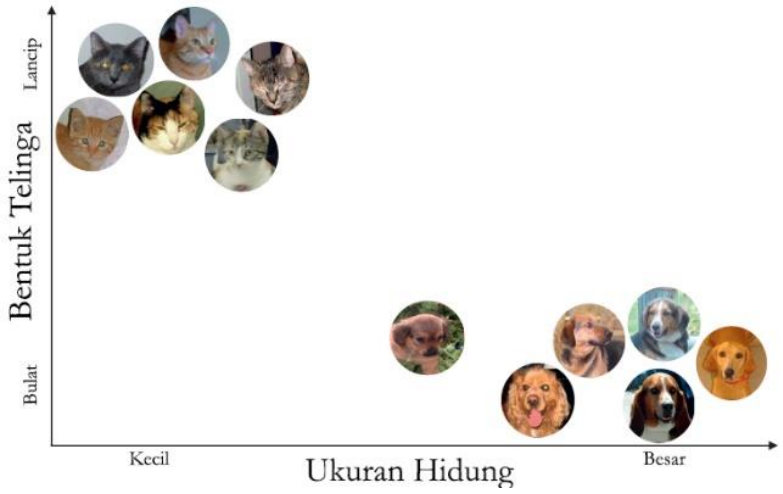
Untuk kasus kucing dan anjing ini akan digunakan dua fitur untuk mempermudah proses pembelajaran dan visualisasi. Dua fitur tersebut adalah **ukuran hidung** dan **bentuk telinga**. Nilai ukuran hidung beragam antara kecil sampai besar, sedangkan bentuk telinga memiliki nilai dari bulat

hingga lancip. Hasil pengukuran telinga dan hidung beberapa ekor kucing dan anjing dapat dilihat pada Tabel 1.1.

Tabel 1.1. Data sampel kucing dan anjing (telinga dan hidung)

Gambar	Bentuk Telinga	Ukuran Hidung	Kelas
	Lancip (3.8 cm)	Kecil (1.2 cm)	Kucing
	Lancip (4 cm)	Kecil (1.5 cm)	Kucing
	Lancip (3.35 cm)	Kecil (1.6 cm)	Kucing
	Lancip (3.5 cm)	Kecil (1.4 cm)	Kucing
	Lancip (3.4 cm)	Kecil (1 cm)	Kucing
	Lancip (3.6 cm)	Kecil (1.65 cm)	Kucing
	Bulat Lonjong (6 cm)	Besar (3 cm)	Anjing
	Bulat Lonjong (6 cm)	Besar (2.3 cm)	Anjing
	Bulat Lonjong (6 cm)	Besar (3.5 cm)	Anjing
	Bulat Lonjong (7 cm)	Besar (2.5 cm)	Anjing
	Bulat Lonjong(6.5 cm)	Besar (3.8 cm)	Anjing
	Bulat Lonjong (7 cm)	Besar (3.5 cm)	Anjing

Berdasarkan data dari Tabel 1.1dilakukan pemetaan kedalam ruang dua dimensi (2-*dimensional feature space*) agar mempermudah visualisasi fitur-fitur maka fitur ukuran hidung sebagai sumbu horizontal dan bentuk telinga sebagai sumbu vertikal seperti yang terlihat pada Gambar 1.8.



Gambar 1.8. Pemetaan *data training* pada *feature space* 2 dimensi.

3. Pelatihan Model.

Pada Gambar 1.8 telah dapat dilihat bahwa setiap data telah dipetakan dalam ruang 2 dimensi, langkah selanjutnya adalah pelatihan model berdasarkan *data sample* tersebut. Secara visual terlihat bahwa data kucing dan anjing mengelompok pada posisi tertentu karena pemetaan dari fitur-fitur tadi. Pelatihan bertujuan untuk menemukan sebuah model linier yang dapat memisahkan kedua kelompok tersebut. Model linier dapat dianalogikan sebagai sebuah garis yang memisahkan antara anjing dan kucing pada Gambar 1.8.

Gambar 1.9 adalah model yang dihasilkan berupa sebuah fungsi garis yang membatasi anggota-anggota kelas kucing dan kelas anjing. Ketika garis ini telah ditentukan maka, semua fitur yang berada di area berwarna merah akan dianggap masuk ke kelas kucing sedangkan yang berada di bawah baris (area berwarna biru) akan dianggap masuk ke kelas anjing. Kesimpulannya, proses *training* adalah proses menemukan fungsi pembatas, pada kasus ini garis pembatas antara anggota-anggota kucing dan anjing. Dengan memanfaatkan pembatas tersebut kita telah dapat membedakan mana anggota kucing dan anggota anjing berdasarkan posisi ketika seekor

makhluk (yang belum diketahui kucing atau anjing) tersebut dipetakan ke diagram atau model.

Gambar 1.9. Ilustrasi hasil pelatihan model menggunakan fungsi linear

4. Uji coba model.

Proses ini bertujuan untuk menguji apakah model yang telah kita hasilkan memberikan hasil yang akurat. Untuk menguji diperlukan sejumlah data yang biasa disebut data uji. Data uji tersebut akan digunakan oleh model untuk memprediksi label. Akurasi atau performa akan dihitung dengan membandingkan label sebenar dan label hasil prediksi.

Gambar 1.10. Sekumpulan data kucing (kotak kiri) dan data anjing (kotak kanan) yang akan digunakan sebagai data uji

Pada Gambar 1.10 berisi sejumlah data kucing dan anjing baru yang belum pernah diidentifikasi oleh sistem. Langkah pertama yang harus

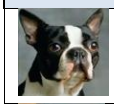
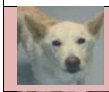
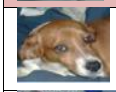
dilakukan adalah melakukan penghitungan terhadap fitur (mengukur ukuran hidung dan bentuk telinga), lalu memasukkan ke dalam model. Setelah dipetakan menggunakan model, kita dapat mengetahui masing-masing data berada di area merah atau biru (perhatikan Gambar 1.11)

Gambar 1.11. Sekumpulan data kucing (kotak kiri) dan data anjing (kotak kanan) yang akan digunakan sebagai data uji

Anjing dengan ID T02 (perhatikan Tabel 1.2 Data uji fitur kucing dan anjing (bentuk telinga dan ukuran hidung)) diklasifikasikan sebagai kucing karena memiliki telinga yang lancip dan hidung sedang. Hal ini merupakan salah satu bentuk eror pada *Machine Learning*. Error ini terjadi dapat sebabkan kesalahan pada pemilihan fitur atau *data training* yang kurang lengkap atau metode pelatihan yang digunakan. Pada *data training* semua anjing memiliki telinga yang lonjong dan hidung yang kecil. Ketika ada anjing yang memiliki telinga lancip maka model yang telah dihasilkan salah dalam mengklasifikasikan. Untuk menghasilkan model yang bagus maka pemilihan fitur dan keberagaman dan jumlah *data training* juga memiliki pengaruh terhadap performa model.

Untuk meningkatkan performa algoritma maka kita harus mengumpulkan lebih banyak data dan fitur pembeda lainnya misalnya jenis ekor, warna mata atau tinggi leher. Setelah data dikumpulkan maka kita latih lagi modelnya untuk menemukan garis pembatas yang baru.

Tabel 1.2 Data uji fitur kucing dan anjing (bentuk telinga dan ukuran hidung)

Gambar	ID	Bentuk Telinga	Ukuran Hidung	Kelas Sebenar	Kelas Model
	T01	Lancip	Besar		
	T02	Lancip	Sedang	Anjing	Kucing
	T03	Bulat Lonjong	Besar	Anjing	Anjing
	T04	Lancip	Kecil	Kucing	Kucing
	T05	Lancip	Kecil	Kucing	Kucing
	T06	Lancip	Kecil	Kucing	Kucing

Klasifikasi error, kelas sebenar adalah anjing, tetapi dideteksi sebagai kucing oleh model yang telah dilatih

1.5 Klasifikasi *Machine Learning*

Machine learning dapat dikelompokkan berdasarkan bagaimana cara belajar sehingga dapat melakukan tugasnya. Pembagian *Machine learning* berdasarkan cara belajarnya dibagi menjadi tiga kelompok yaitu :

1. Supervised Learning

Secara bahasa, *supervised learning* adalah pembelajaran terarah/terawasi. Jika kita analogikan pada proses pembelajaran, komputer atau mesin akan mempelajari *data training* yang berisi label. Jika dianalogikan ke siswa dan guru, dimana komputer sebagai siswa yang belajar maka guru akan meminta siswa untuk belajar dari soal yang sudah memiliki solusi dan kunci jawaban.

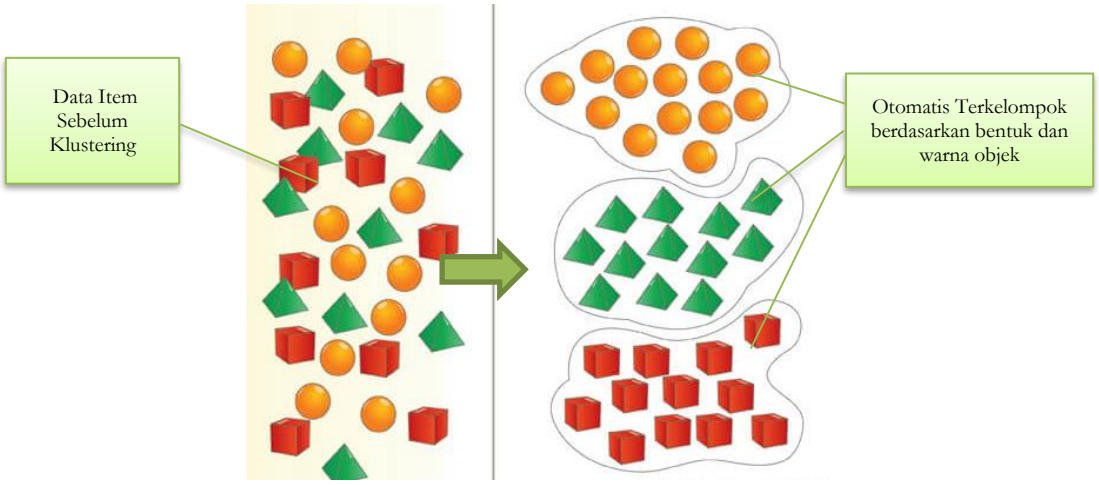
Contohnya kasus pendeteksian kucing dan anjing tadi, ketika kita memberikan data kucing dan anjing sebagai *data training*, sama artinya kita menginformasikan komputer tentang contoh data kucing dan data anjing. Kemudian komputer mempelajarinya dengan algoritma tertentu, lalu membuat model berdasarkan *data training* yang berisi contoh data anjing dan kucing. Hasilnya berdasarkan contoh yang telah diberikan, komputer dapat mengelompokkan kucing dan anjing yang belum pernah dilihat. Pada intinya ketika melakukan proses pembelajaran, sistem diberikan data yang berisi solusi atau hasil yang akan diinginkan atau contoh kasus yang akan diselesaikan oleh sistem.

Selain klasifikasi, regresi juga dapat dimasukkan sebagai pembelajaran terarah. Adapun contoh-contoh algoritma yang termasuk ke dalam kategori supervised adalah *Linear Regression* (Regresi Linear), *Logistic Regression* (Regresi Logistik), *Linear Discriminant Analysis*, *k-Nearest Neighbors*, *Support Vector Machines* (SVMs), *Random Forests*, dan *Neural networks*.

2. Unsupervised Learning

Berbeda dengan *supervised learning*, *unsupervised learning* adalah kebalikannya dimana proses pembelajaran dilakukan tanpa petunjuk. Algoritma dalam komputerlah yang bekerja untuk menemukan pola-pola di dalam data. Secara matematis, *unsupervised learning* terjadi ketika kita memiliki sejumlah data masukan (X) dan tanpa variabel *output* yang berhubungan. Jika kita menggunakan analogi yang sama dengan siswa tadi maka, siswa belajar tanpa ada solusi dan jawaban yang benar, siswa harus menemukan sendiri jawabannya. Masalah *Unsupervised learning* dapat dibagi menjadi dua jenis yaitu asosiasi dan *clustering*

Sebagai contoh, pada Gambar 1.12 berisi dua jenis item yang berbeda. Kedua item tersebut akan dipisah menjadi beberapa kategori tergantung data. Komputer hanya mengetahui fitur-fitur yang akan digunakan untuk membedakan kedua item tersebut yaitu warna dan bentuk. Dengan menggunakan algoritma *clustering*, komputer akan dapat membagi item-item menjadi dua kelompok tanpa harus diberi pengetahuan. Algoritma akan bekerja untuk membagi menjadi beberapa kelompok dengan melihat isi data masing-masing item. Teknik-teknik mengelompokkan ini akan dibahas pada bab-bab berikutnya. Contoh algoritma *unsupervised learning* sederhana adalah K-means.



Gambar 1.12. Ilustrasi proses klusterisasi data

3. Reinforcement Learning

Sebuah komputer akan berinteraksi dengan sebuah lingkungan yang sangat dinamis dimana komputer tersebut harus melakukan sebuah tugas tertentu contohnya pada permainan catur atau *self-driving car*. Melalui sebuah algoritma, mesin akan mempelajari bagaimana membuat keputusan yang spesifik berdasarkan lingkungan yang berubah-ubah.

Selain berdasarkan cara belajar, *machine learning* dapat dikategorikan berdasarkan proses pembelajaran dilakukan secara bertahap (***Batch Learning***)atau secara langsung (*on the fly* - ***Online Learning*).** Prinsip dasar dari *online learning* adalah menghasilkan sebuah model yang melakukan proses pembelajaran data baru secara *realtime* atau mendekati *realtime*. Sedangkan *Batch Learning*, *data training* akan dipecah-pecah menjadi beberapa bagian lalu setiap bagiannya akan dipelajari secara terpisah pada waktu yang berbeda.

Machine learning berdasarkan bagaimana cara kerjanya dapat dikolompokkan menjadi dua yaitu

- ***Instance-based learning*** (atau sering disebut *memory-based learning*) adalah sebuah kelompok algoritma ML yang bekerja dengan membandingkan data testing dengan data yang telah dipelajari pada proses training. Algoritma kelompok ini tidak membuat sebuah generalisasi tetapi lebih ke arah perbandingan dengan data yang disimpan di memori. Contoh algoritma yang masuk ke kelompok ini adalah *k-nearest neighbors, kernel machines*, dan RBF network.
- ***Model based learning*** adalah kebalikan dari intance-based dimana menggunakan memori untuk melakukan pemecahan masalah, algoritma ini membuat sebuah model yang bersifat generik.

Kriteria-kriteria ini tidak eksklusif, kita dapat mengombinasikannya. Sebagai contoh sebuah sistem Anti-Spam yang dapat *mempelajari spam-on-the-fly* menggunakan *deep neural network* dilatih menggunakan *data training*

spam dan ham; maka sistem ini dikatakan Online, *model-based* dan *supervised learning system*.

1.6 Algoritma Machine Learning

Agar mempermudah dalam penggunaannya maka algoritma ML perlu dikelompokkan berdasarkan permasalahan yang ingin dipecahkan. Adapun pembagian tersebut adalah

1.6.1 Algoritma Regresi

Permasalahan Regresi fokus kepada pembuatan model yang memetakan variabel independen ke variabel dependen. Biasanya algoritma ini digunakan untuk peramalan atau klasifikasi. Penjelasan lebih lanjut tentang materi ini ada di bab selanjutnya. Adapun algoritma yang popular pada kelompok ini adalah *Ordinary Least Squares Regression* (OLSR), *Regresi Linear, Regresi Logistik, Regresi Stepwise, Multivariate Adaptive Regression Splines* (MARS), *Locally Estimated Scatterplot Smoothing* (LOESS).

1.6.2 Algoritma Regularisasi

Algoritma regularisasi merupakan ekstensi dari metode lainnya (biasanya metode regresi). Manfaat dari algoritma ini adalah dapat mencegah *overfitting* (akan dibahas pada bab 3) dengan menyederhanakan model yang dihasilkan sehingga menghasilkan generalisasi yang lebih baik. Beberapa algoritma regularisasi yang memiliki performa bagus adalah *Least Absolute Shrinkage and Selection Operator* (LASSO) atau L1 *Regularization*, *Ridge Regression* atau L2 *Regularization*, Elastic Net dan *Least-Angle Regression* (LARS)

1.6.3 Algoritma Instance-based Learning

Model pembelajaran berbasis instance (*Instance-based Learning*) sederhananya adalah sebuah model dimana dalam pengambilan keputusan berdasarkan contoh atau data pelatihan. Algoritma ini menilai kemiripan antara contoh dan masalah, sehingga hasil yang didapatkan adalah yang paling mirip dengan contoh. Pada metode seperti ini biasanya dibangun sebuah basis data contoh dan membandingkan data baru dengan database

menggunakan ukuran kesamaan untuk menemukan yang paling cocok dan membuat prediksi. Adapun algoritma-algoritma yang popular adalah *k-Nearest Neighbor* (KNN), *Learning Vector Quantization* (LVQ), *Self-Organizing Map* (SOM), *Locally Weighted Learning* (LWL) dan lain-lain.

1.6.4 Algoritma Decision Tree

Algoritma *Decision Tree* atau pohon keputusan adalah metode-metode yang digunakan untuk membuat sebuah model pengambilan keputusan berbentuk *tree*/pohon berdasarkan atribut *data training*. Berikut ini adalah beberapa algoritma yang populer *Classification and Regression Tree* (CART), *Iterative Dichotomiser 3* (ID3), C4.5 dan C5.0

1.6.5 Algoritma Bayesian

Algoritma Bayesian adalah metode-metode yang secara explisit menggunakan Teorema Bayes untuk memecahkan masalah klasifikasi dan regresi. Berikut beberapa metode yang masuk ke dalam kategori ini Naive Bayes, Gaussian Naive Bayes, *Multinomial Naive Bayes*, *Bayesian Belief Network* (BBN).

1.6.6 Algortima Klustering

Clustering atau klasterisasi adalah metode pengelompokan data. Menurut Tan, 2006 *clustering* adalah sebuah proses untuk mengelompokkan data ke dalam beberapa kluster atau kelompok sehingga data dalam satu kluster memiliki tingkat kemiripan yang maksimum dan data antar cluster memiliki kemiripan yang minimum. Berikut ini beberapa algoritma klasterisasi yang populer k-Means, k-Medians, *Expectation Maximisation* (EM), *Agglomerative Nesting* (AGNES), *Divisive Analysis* (DIANA)

1.6.7 Algortima Association Rule Learning

Association rule adalah metode yang mengekstrak *rule* sehingga didapatkan relasi di antara variabel pada data. Adapun algoritma yang popular adalah Apriori dan Eclat

1.6.8 Algoritma Artificial Neural Network

Artificial Neural Networks (Jaringan Syaraf Tiruan) adalah metode yang membuat model yang diinspirasi dari struktur jaringan syaraf. Beberapa algoritma yang populer adalah *Perceptron, Back-Propagation, Hopfield Network, Radial Basis Function Network* (RBFN).

1.6.9 Algoritma Deep Learning

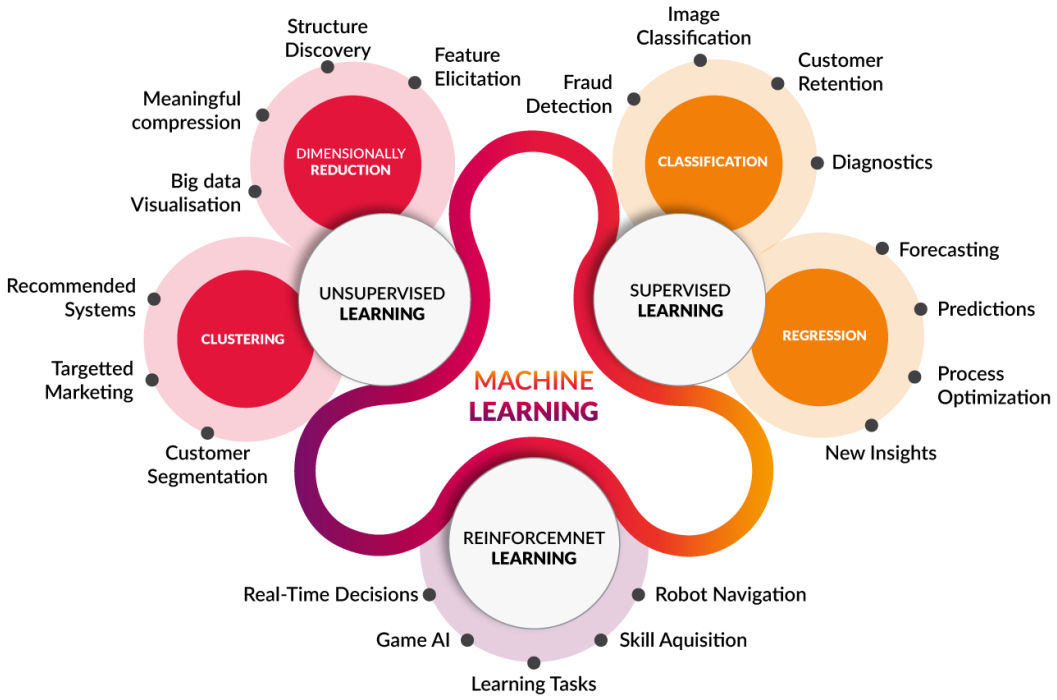
Deep Learning adalah pengembangan dari Neural Network dengan jaringan yang lebih kompleks. Berikut beberapa algoritma yang populer pada kategori ini. *Convolutional Neural Network (*CNN*, Deep Boltzmann Machine (*DBM*), Deep Belief Networks (*DBN*), Stacked Auto-Encoders, Recurent Neural Network (RNN)* dan lain-lain

1.6.10 Algoritma Ensemble

Metode Ensemble atau metode ansambel adalah algoritma dalam pembelajaran mesin dimana algoritma ini sebagai pencarian solusi prediksi terbaik dibandingkan dengan algoritma yang lain karena metode ensambel ini menggunakan beberapa algoritma pembelajaran untuk pencapaian solusi prediksi yang lebih baik. Berikut ini beberapa algoritma ensambel yang populer *Boosting, Bootstrapped Aggregation* (Bagging), AdaBoost, *Gradient Boosting Machines* (GBM), *Gradient Boosted Regression Trees* (GBRT) dan *Random Forest*

1.7 Aplikasi Machine Learning

Pemanfaatan aplikasi *Machine Learning* sudah terasa dalam kehidupan sehari-hari. Kegiatan-kegiatan yang seperti sistem rekomendasi, prediksi, peramalan, diagnosa, robot canggih telah memanfaatkan teknologi ML. Contoh mudahnya adalah produk-produk Google, misalnya Google Translate (*machine translation, handwritten recognition, speech recognition, Alpha Go*), Google Search dan Gmail.



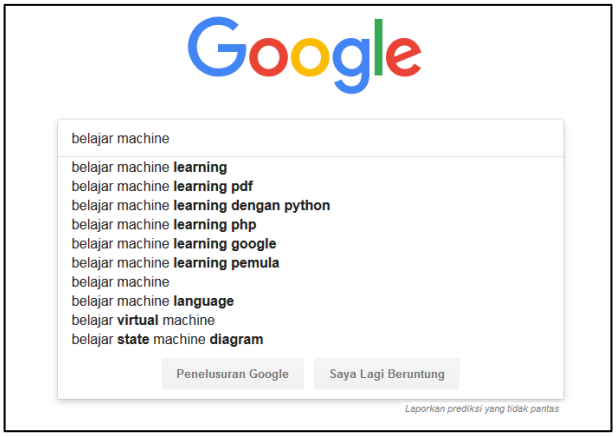
Gambar 1.13. Masalah yang dapat diselesaikan dengan ML(sumber: Medium.com)

Berikut ini beberapa contoh pengaplikasian ML dalam kehidupan sehari-hari:

1. Mesin Pencari

Seluruh mesin pencari seperti Google, Bing dan Yandex menerapkan *machine learning* untuk melakukan perangkingan laman suatu website, rekomendasi *keyword*, perbaikan jika ada pengejaan dan rekomendasi

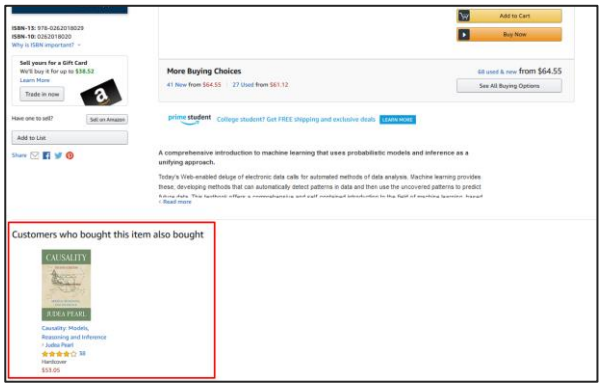
keyword yang lebih baik. Setiap mesin pencari mempunyai resep tersendiri pada algoritma pencarian yang digunakannya. Dalam praktiknya ketika kita mengetik kata kunci, Google akan menampilkan hasil pencarian yang paling mendekati kata kunci tersebut. Apabila kita memilih suatu halaman dan menghabiskan banyak waktu pada halaman tersebut, Google akan mendeteksi bahwa halaman tersebut sesuai dengan kata kunci yang kita masukkan. Begitu pula, saat kita melihat halaman pencarian berikutnya misalnya halaman 2, 3, dan seterusnya. Google akan mendeteksi adanya ketidaksesuaian kata kunci dengan hasil pencarian yang dihasilkan. Begitulah data tersebut terkumpul dan diakumulasikan menggunakan *Machine Learning* oleh Google Search Engine, untuk menghasilkan hasil pencarian yang dinamis dan berkualitas.



Gambar 1.14. Rekomendasi kata kunci pencarian Google

2. Toko Online dan Marketplace

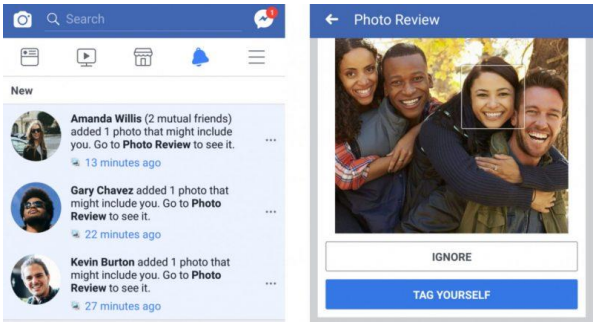
Penggunaan *Machine Learning* di *marketplace* atau toko Online dapat menguntungkan penjual dan pembeli. Untuk meningkatkan pengalaman pengguna dan meningkatkan penjualan, setiap akun perlu menampilkan rekomendasi produk yang sesuai dengan minat pembelian pembeli. Untuk melakukan ini secara otomatis dan *real-time*, tentunya *Machine Learning* sangat menentukan keakuratan rekomendasi produk tiap pembeli di akunnya. Sebagai contoh, ketika kita belanja buku di Amazon maka kita akan direkomendasikan buku yang dibeli bersamaan dengan buku yang kita cari.



Gambar 1.15. Rekomendasi kata kunci pencarian Google

3. Media Sosial

Media Sosial adalah salah satu aplikasi yang paling banyak digunakan oleh masyarakat, oleh karena itu provider layanan media sosial selalu menggunakan fitur yang canggih untuk membuat penggunanya merasa mudah, aman dan nyaman. Sebagai contoh sosial media facebook, ia menggunakan beberapa teknologi ML untuk mempermudah pengguna diantaranya adalah rekomendasi teman yang mungkin kita kenal, pengenalan wajah, urutan konten yang ditampilkan pada beranda dan lain-lain.



Gambar 1.16. Facebook mengenali bagian wajah

4. Periklanan Digital

Google Adword adalah media periklanan digital terpopuler di dunia. Adword menampilkan iklan-iklan pada situs web yang menjadi *publiser* di Google Adsense. Iklan yang ditampilkan adalah iklan yang bersifat dinamis

atau berubah-ubah. Google Adword mengumpulkan data situs berdasarkan topiknya, kemudian menampilkan iklan-iklan yang relevan dengan topik tersebut. Di samping itu Google Adword juga menggunakan cookies, sebagai referensi aktivitas pengunjung suatu website terkait situs-situs yang dikunjungi sebelumnya. Adword melakukan akumulasi terhadap 2 faktor tersebut, sehingga dapat ditampilkan iklan yang sesuai dengan pengunjung website.

5. Asisten Pribadi Virtual

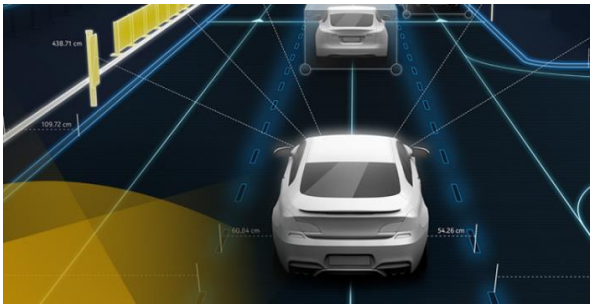
Gadget telah dilengkapi dengan asisten pribadi virtual, baik laptop maupun smartphone, misalnya: Cortana di Microsoft Windows, Siri di Iphone, dan Google Now di Android. Asisten virtual ini dapat membantu penggunanya untuk melakukan pencarian di internet, menanyakan jalan, cuaca, melakukan panggilan telepon, hingga membuka aplikasi. Di Windows, Cortana bahkan mempelajari penggunanya untuk memberikan rekomendasi perintah yang akan dilakukan.

6. Penyaring Pesan Spam

Machine learning memegang andil yang sangat besar untuk melakukan penyaringan (*filter*) spam baik di email, website, hingga media komunikasi berbasis software terinstall. Algoritma pohon keputusan (*decision tree*) merupakan cikal bakal dari algoritma *spam filtering*, untuk mentukan suatu pesan termasuk spam atau bukan.

7. **Mobil Kendali Otomatis**

Mobil kendali otomatis merupakan penerapan serta pengembangan dari machine learning yaitu machine vision. Mobil kendali otomatis merupakan penerapan machine learning yang kompleks dan dengan resiko langsung yang tinggi. Banyak hal yang harus dipelajari oleh mobil, mulai dari rambu-rambu lalu lintas, arah dan tujuan, kondisi jalan, *traffic light*, kondisi manusia sekitarnya, dan sensor lainnya yang terintegrasi.



Gambar 1.17. Implementasi mobin kendali otomatis (sumber:Lecun.com)

1.8 Latihan

1. Jelaskan pengertian *Machine Learning*
2. Jelaskan pembagian ML berdasarkan *Machine learning* berdasarkan cara belajarnya?
3. Apakah perbedaan antara *Machine Learning*, Kecerdasan Buatan, dan Data Mining?
4. Sebutkan Aplikasi-aplikasi ML lainnya selain yang telah disebutkan?

Bab

2

Teori Pendukung *Machine Learning*

Sabar Adalah Bahan Ramuan Paling Sehat dalam Kehidupan – **Umar bin Khattab**.

Capaian Pembelajaran:

1. Mampu mengetahui dan memahami teori-teori pendukung *Machine Learning* seperti komputasi, matematika dan statistika
2. Mampu mengimplementasikan teori-teori pendukung tersebut menggunakan bahasa Python

Machine learning (ML) sebuah cabang ilmu yang dibangun oleh ilmu-ilmu lain seperti Aljabar Linier, Kalkulus dan Statistika. Ilmu-ilmu dasar tersebut sangat dibutuhkan untuk mengetahui konsep dasar dari ML. Selain memahami konsep dasar tersebut, kemampuan teknis seperti algoritma dan pemrograman juga diperlukan agar kita dapat mengaplikasikan ilmu-ilmu dasar tersebut sehingga dapat bermanfaat. Kemampuan komputasi merupakan peralatan yang dibutuhkan untuk mengoperasikan ML, karena tanpa daya komputasi konsep-konsep tersebut tidak dapat direalisasikan.

Jika kita analogikan, dasar-dasar matematika sebagai otak dan kemampuan komputasi sebagai otot dan badannya yang menggerakkan. Kedua hal tersebut saling melengkapi.

2.1 Kemampuan Komputasi

Kemampuan komputasi identik dengan kemampuan untuk menggunakan bahasa pemrograman atau *tool* berbasis komputer tertentu untuk memproses data. Saat ini banyak aplikasi atau *tool* yang dapat digunakan untuk menyelesaikan permasalahan ML atau pembelajaran mesin. Sederhananya, Anda tinggal memasukkan data lalu aplikasi tersebut akan memproses data Anda lalu menghasilkan model yang dapat digunakan untuk melakukan prediksi. Sebagian besar aplikasi tersebut sudah mengimplementasi algoritma-algoritma yang populer yang dapat menyelesaikan masalah Anda.

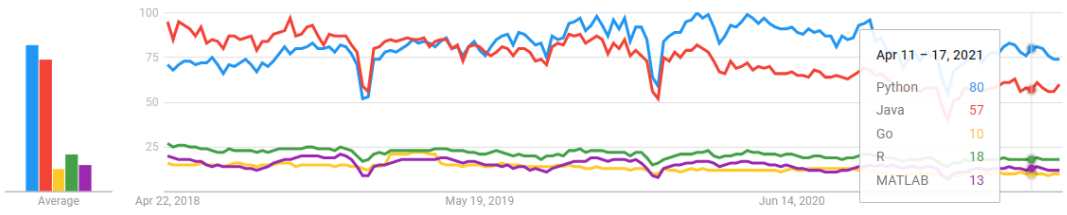
Ibarat seorang mekanik yang akan memperbaiki mobil, maka dia harus dapat memahami permasalahan mobil kemudian menggunakan peralatannya untuk mengidentifikasi permasalahan dan melakukan perbaikan. Semua peralatan akan mempunyai fungsi masing-masing, dan sebuah aktivitas perbaikan terdiri dari serangkaian aktivitas yang harus dilakukan secara berurutan. Kadang-kadang dengan pengalaman yang dimiliki dia melakukan "hacking" terhadap permasalahan kita sehingga permasalahan tersebut dapat dipecahkan.

Hal tersebut hampir sama dengan yang dilakukan oleh programer dalam memecahkan sebuah masalah. Sejumlah kegiatan harus dilakukan agar sebuah kasus dapat terpecahkan. Proses analisa, perancangan dan implementasi menjadi urutan yang tidak dapat dihindari. Oleh karena itu ketika kita mempelajari ML maka kita harus memiliki kemampuan dibidang programming untuk melakukan implementasi dari apa yang telah kita analisis dan rancang.

2.1.1 Python

Python adalah bahasa yang sangat terkenal di antara pengembang *Machine Learning, Data scientists* maupun *Data Miner*. Alasannya karena bahasa ini mudah dan gratis. Selain itu Python juga bahasa pemrograman yang interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, Python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintak. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya. Menurut Google Trend, sampai tahun 2021 perkembangan pengguna Python signifikan dan saat ini merupakan bahasa yang paling banyak dibicarakan.



Gambar 2.1. Perkembangan Python (sumber: google trend, 2021)

Pada buku ini akan menggunakan Python sebagai bahasa yang digunakan ketika implementasi karena kemudahannya. Berbeda dengan bahasa yang populer dibidang sains seperti R dan MATLAB, Python lebih banyak diminati karena ia termasuk bahasa pemrograman yang umum sehingga pengembang dapat membuat aplikasi untuk Enterprise lebih mudah.

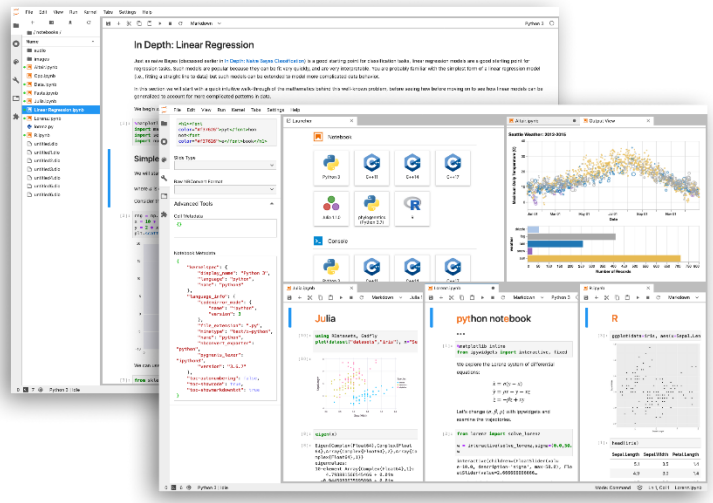
Berikut adalah beberapa kelebihan Python:

- **Mudah dipelajari dan singkat**. Bahasa pemrograman Python memiliki sedikit *keyword* dan struktur yang sederhana sehingga mudah dipelajari bagi pemula. Selain itu, kode program yang dipakai Python jauh lebih singkat dibandingkan dengan bahasa pemrograman lain untuk menyelesaikan masalah yang sama.
- **Mudah di baca**. Python menggunakan *keyword* yang sederhana serta menwajibkan penggunaan spasi atau tab untuk *indentation* kode program (mengatur kode dengan menjorokkan ke dalam sesuai blok), sehingga kode yang ditulis lebih rapi. Dalam bahasa pemrograman lain, *indentation* ini tidak diwajibkan.
- **Cross-platform**. Python bisa dipakai lintas sistem operasi, termasuk Windows, UNIX, Linux, Mac OS, dll. Kode program yang ditulis satu kali, tidak perlu diubah sesuai agar bisa berjalan di sistem operasi yang lain.
- **Mendukung multi-paradigma**. Python dibangun untuk tujuan umum sehingga ditulis dengan menggunakan paradigma pendekatan prosedural maupun pemrograman objek (OOP). Programmer yang tidak terbiasa dengan OOP dapat menggunakan Python secara prosedural.
- **Memiliki banyak library dan package Manager**. Library adalah sebutan untuk kode program tambahan yang dikembangkan oleh orang lain untuk menyelesaikan hal-hal khusus. Python memiliki lebih dari 170.000 *library* atau *packages* yang dikembangkan di komunitas pypi.org. Selain itu untuk mempermudah pengguna, Python memiliki *package system* yang disebut pip. Dengan adanya package manager pengguna dapat mengistal dan mengupdate library dengan mudah tanpa pusing dengan depedensi.
- **Gratis dan opensource**. Python dikembangkan sebagai project open source dan bisa digunakan siapa saja secara gratis

- **Komunitas yang kuat.** Jika menemukan permasalahan, anda dapat mengunjungi stackoverflow dan mendapatkan jawaban dalam waktu singkat

2.1.2 Jupyter Notebook

Jupyter Notebook (dulunya, IPython Notebook) adalah sebuah aplikasi yang umum digunakan pada bidang *Data Science*. Biasanya aplikasi ini digunakan untuk membuat dan berbagi dokumen yang berisi *Live code*, Persamaan (*Equalization*), *Visualizations*, dan teks penjelasan tentang dokumen tersebut. Jupyter Notebook merupakan aplikasi server-client yang mengizinkan kita untuk melakukan *editing* dan *running/compile* dokumen notebook melalui web browser.



Gambar 2.2. Aplikasi Jupyter Notebook

Dengan *tool* ini kita dapat melakukan banyak hal misalnya *data cleaning*, analisa data, pemodelan, bahkan pelatihan dan evaluasi model. Penggunaan Jupyter disukai karena di eksekusi per baris, sehingga kita dapat langsung melihat luaran yang kita inginkan. Jika dibanding kan dengan cara konvensional dimana kita harus mengetik semua kode terlebih dahulu baru dapat menjalankannya, cara ini lebih sederhana dan mudah apalagi bagi yang sedang belajar *Machine Learning*.

2.1.3 NumPy

NumPy (kependekan dari *Numerical Python*) adalah library Python yang fokus pada *scientific computing*. NumPy memiliki kemampuan untuk membentuk objek *N-dimensional array*, yang mirip dengan list pada Python. Keunggulan NumPy array dibandingkan dengan list pada Python adalah konsumsi memory yang lebih kecil serta *runtime* yang lebih cepat. Numpy mempermudah untuk melakukan perhitungan larik multi-dimensi (*multi-dimensional arrays*) dan perhitungan matriks. Selain penggunaannya dalam menyelesaikan persamaan aljabar linier (*linear algebra equations*) dan perhitungan matematis lainnya, NumPy juga digunakan sebagai wadah multi-dimensi serbaguna untuk berbagai jenis data generik.

Salah satu lain kenapa Numpy sering digunakan adalah List pada Python tidak mendukung penuh kemudahan *scientific computing*, sebagai contoh kita akan lakukan operasi penjumlahan pada 2 list seperti berikut ini:

```
[1]  1  #Penjumlahan dua List Python
     2  a = [1,2,3]
     3  b = [4,5,6]
     4
     5  print('a+b = ', a+b)
     6
```

```
a+b =  [1, 2, 3, 4, 5, 6]
```

Ketika kita ingin menjumlahkan tiap elemen pada list a ([1,2,3]) dan list b ([4,5,6]) dengan operator +, maka hasilnya adalah penggabungan (*concat*) keduanya yaitu [1,2,3,4,5,6]. Padahal pada *scientific computing,* operator + bertujuan untuk melakukan penjumlahan terhadap kedua list tersebut. Oleh karena itu, kita harus menggunakan perulangan untuk menambahkan tiap elemen pada list a dan list b. Proses penjumlahan list yang menggunakan perulangan *for* membutuhkan waktu yang lama dan tidak efisien dari sisi penulisan code.

Lebih hebatnya, NumPy terintegrasi dengan bahasa pemrograman lain seperti C / C ++ dan Fortran. Fleksibilitas perpustakaan NumPy

memungkinkannya untuk dengan mudah dan cepat bergabung dengan berbagai *database* dan *tools*. Sebagai contoh, mari kita lihat bagaimana NumPy (disingkat np) dapat digunakan untuk mengalikan dua matriks.

Untuk dapat menggunakan Numpy maka harus dilakukan import terhadap library tersebut. Berikut contoh melakukan import terhadap library NumPy

```
[3]  1  #inisialiasi penggunaan Numpy
     2  import numpy as np
```

Setelah melakukan import, maka kita dapat menggunakan library tersebut. Sebagai contoh pertama kita akan membuat sebuah array (vektor) `[1, 2, 3]` dan matrix `[[6,5],[11,7],[4,8]]`.

```
[4]  1  #membuat sebuah array
     2  arr_1_dimensi = np.array([1, 2, 3])
     3  arr_2_dimensi = np.array([[6, 5], [11, 7], [4, 8]])
     4
     5  print("Ukuran Array 1 Dimensi : ", arr_1_dimensi.shape)
     6  print("Ukuran Array 2 Dimensi : ", arr_2_dimensi.shape)
     7
```

```
Ukuran Array 1 Dimensi :  (3,)
Ukuran Array 2 Dimensi :  (3, 2)
```

Pada input baris ke dua, kita membuat sebuah vektor menggunakan method `np.array`, yang membutuhkan sebuah parameter berupa list. List tersebut akan dikonversi menjadi array numpy lalu kita akan menggunakan atribut shape pada baris ke lima untuk melihat dimensi dari vektor atau array 1 dimensi yang telah kita buat. Sedangkan pada paris kedua kita membuat sebuah matrik 3 x 2 (3 baris dan 2 Kolom) menggunakan perintah yang sama tetapi dengan list yang berbeda sehingga kita dapat membuat sebuah array dua dimensi seperti berikut ini.

$$\text{array\_2\_dimensi} = \begin{bmatrix} 6 & 5 \\ 11 & 7 \\ 4 & 8 \end{bmatrix}$$

Pada proses pembuatan array dilakukan *upcasting* ketika tipe data elemen array tidak sama, dilakukan penyamaan tipe data pada yang lebih tinggi. Sebagai contoh kita membuat array numerik dengan semua elemen bertipe integer, kecuali 1 elemen bertipe float (ex: `[1,2,3.2]`), maka otomatis akan dilakukan *upcasting* menjadi tipe float pada semua elemen array yaitu `[1.,2.,3.2]`.

Selain `np.array` , kita juga dapat menggunakan metode lain untuk membuat sebuah array seperti `zeros`, `ones`, `arange` dan lain-lain. Metode `zeros` dan `ones` membutuhkan parameter yang sama yaitu sebuah integer atau tuple integer yang menunjukkan ukuran array yang akan digunakan contoh (2, 3) atau 2. Method `zeros` akan menghasilkan array yang bernilai 0 sedangkan method `ones` akan menghasilkan array yang bernilai 1.

```
[5]  1  #membuat array  dengan ukuran 2 x3 berisi 0
     2  c = np.zeros((2, 3))
     3  #membuat array  dengan ukuran 5 x5 berisi 1
     4  d = np.ones((5, 6))
     5
     6  print(c)
     7  print(d)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]]
```

Selain metode tersebut, metode `arange` juga merupakan metode yang sering digunakan. Prinsip kerjanya mirip dengan `range`, dimana kita dapat membuat array dari *m* ke *n-1* dengan nilai penambahan *o*, *m* adalah nilai awal dan *n* adalah nilai akhir dan *o* adalah nilai penambahan. Misalnya jika ingin membuat array dengan isi [0,1,2,3,4,5,6,7,8,9], maka *m*=0 , *n*=10, *o*=1. Jika *o* tidak disebutkan maka secara default bernilai 1, dan *m* bernilai 0. Perhatikan contoh berikut:

```
[6]  1  c = np.arange(0, 10)
     2  d = np.arange(0, 10, 2)
     3
     4  print(c)
     5  print(d)
```

```
[0 1 2 3 4 5 6 7 8 9]
[0 2 4 6 8]
```

Berikut ini beberapa method yang sering digunakan dalam pengolahan data untuk pelatihan ataupun pengujian.

- Mengetahui ukuran array. Ukuran array menunjukkan panjang atau jumlah elemen yang berada didalam array tersebut.

```
[9]  1  numpy_array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
     2  print(numpy_array.size)
```

```
9
```

- Mengetahui Dimensi array. Untuk mendapatkan dimensi sebuah array dapat menggunakan attribut shape.

```
[10]  1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix.shape)
```

```
(2, 4)
```

- Mengubah dimensi array. Perubahan dimensi dapat dilakukan dengan menggunakan fungsi reshape. Perhatikan contoh input ke-14, dimana mengubah array 2x4 menjadi array 4x2.

```
[11]  1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  numpy_matrix = numpy_matrix.reshape(4, 2)
      3  print(numpy_matrix)
```

```
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

- Mengambil sebuah nilai pada posisi tertentu. Untuk mengambil nilai pada lokasi tertentu cukup dengan memberikan alamat index baris dan kolom, dimana index dimulai dari nol. Contoh array [0,2] akan mengambil pada baris ke 1 kolom ke 3

```
[12]  1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix[0, 2])
```

```
3
```

- Mengambil beberapa nilai dari array: Pengambilan beberapa nilai mirip dengan pengambilan sebuah nilai, bedanya adalah kolom atau baris dalam bentuk index. Contohnya [0, 0:4] akan menghasilkan nilai pada baris pertama dan kolom dengan indek 0 sampai 3

```
[14]  1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix[0, 0:3])
```

```
[1 2 3]
```

- Membalikkan Array dapat dilakukan dengan menggunakan " ::-1 " pada posisi baris akan membalik urutan, jika berada dikolom akan membalikkan kolom.

```
[16]  1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix[ 0, ::-1])
```

```
[4 3 2 1]
```

- Mencari nilai maximum, minimum dan total dari array dapat dilakukan dengan mudah. Dari sebuah objek numpy telah disediakan fungsi max untuk menghitung nilai tertinggi, min untuk menghitung nilai terendah dan sum menghitung penjumlahan semua item pada array.

```
[20]  1  numpy_array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
      2  print("MAX :",numpy_array.max())
      3  print("MIN :",numpy_array.min())
      4  print("SUM :",numpy_array.sum())
```

```
MAX : 9
MIN : 1
SUM : 45
```

- Mencari rata, median, varian dan standar deviasi dari array dapat dilakukan dengan memanggul fungsi mean, median, var dan std pada objek array Numpy.

```
[22]  1  numpy_array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
      2  print("MEAN :",numpy_array.mean())
      3  print("MEDIAN :",np.median(numpy_array))
      4  print("VAR    :",numpy_array.var())
      5  print("STD    :",numpy_array.std())
```

```
MEAN : 5.0
MEDIAN : 5.0
VAR    : 6.666666666666667
STD    : 2.581988897471611
```

- Melakukan operasi pada level kolom atau baris. Penggunaan parameter "axis = 0 " akan melakukan proses hanya pada sumbu y sedangkan " axis = 1 " akan melakukan operasi pada sumbu x. " axis = None" akan melakukan operasi pada sumbu x dan y

```
[25]  1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix)
      3  print("SUM[axis=0] : ", numpy_matrix.sum(axis = 0))
      4  print("SUM[axis=1] : ", numpy_matrix.sum(axis = 1))
      5
```

```
[[1 2 3 4]
 [5 6 7 8]]
SUM[axis=0] :  [ 6  8 10 12]
SUM[axis=1] :  [10 26]
```

- Menggabungkan beberapa array. Fungsi concatenate dapat digunakan untuk menggabungkan beberapa array menjadi sebuah array. Sebagai contoh array [1,2,3], [4,5,6], [7,8,9] ketika digabungkan akan menghasilkan [1 2 3 4 5 6 7 8 9]. Sedangkan untuk array yang multi-dimensi [[1,2,3],[0,0,0]] dan [[0,0,0],[7,8,9]] akan menghasilkan [[1 2 3 0 0 0], [0 0 0 7 8 9]]

```
[26]  1  array_1 = np.array([1,2,3])
      2  array_2 = np.array([4,5,6])
      3  array_3 = np.array([7,8,9])
      4  print("1D :",np.concatenate((array_1, array_2, array_3)))
      5  array_4 = np.array([[1,2,3],[0,0,0]])
      6  array_5 = np.array([[0,0,0],[7,8,9]])
      7  print("n-D :",np.concatenate((array_4, array_5), axis = 1))
```

```
1D : [1 2 3 4 5 6 7 8 9]
n-D : [[1 2 3 0 0 0]
 [0 0 0 7 8 9]]
```

Penggunaan Numpy untuk pengolahan matriks dan tensor akan dibahas lebih lanjut pada bagian aljabar linier.

2.1.4 Pandas

Pandas adalah sebuah librari yang menyediakan struktur data dan fungsi analisis data yang mudah digunakan dan memiliki kinerja yang tinggi. Sama seperti NumPy, Pandas juga dikembangkan oleh SciPy dan berlisensi BSD. Pandas menawarkan sekumpulan method/class yang serbaguna dan bermanfaat untuk membuat struktur data dan melakukan analisis data yang besar. Library ini juga dapat mengatasi permasalahan umum pada data seperti data yang tidak lengkap, tidak terstruktur, dan tidak teratur — dan dilengkapi dengan *tool* untuk membentuk, menggabungkan, menganalisis, dan memvisualisasikan datasets.

Pandas memadukan library NumPy yang memiliki kemampuan manipulasi data yang fleksibel dengan database relasional (seperti SQL). Sehingga memudahkan kita untuk melakukan *reshape*, *slice* dan *dice*, agregasi data, dan mengakses subset dari data.

Untuk memulai, lakukan import terlebih dahulu library pandas as pd. Penggunaan "as" di sini, artinya kita menggantikan pemanggilan pandas dengan prefix pd untuk proses berikutnya

```
[1]   1   import pandas as pd
```

Ada tiga jenis struktur data dilibrary ini:

- **Series**: single-dimensional, array homogen

Series merupakan struktur data dasar dalam Pandas. Series bisa juga diibaratkan sebagai array satu dimensi seperti halnya yang ada pada numpy array, hanya bedanya mempunyai index dan kita dapat mengontrol index dari setiap elemen tersebut.

Struktur data yang bisa ditampung berupa integer, float, dan juga string. Series juga mendukung operasi vektor. Secara definisi, Series tidak dapat mempunyai kolom ganda, untuk masalah ini bisa menggunakan struktur data **frame**.

Bila merujuk pada dokumentasi pandas, terdapat struktur dari class Series seperti berikut:

```
pandas.Series(data=None, index=None, dtype=None, name=None,
copy=False, fastpath=False)
```

Adapun parameter-parameter yang dibutuhkan adalah :

- Parameter data bisa berisi array, Iterable, dict, atau nilai scalar
- Parameter index disarankan bernilai unik dan hashable yang memiliki panjang sama dengan data
- Parameter dtype menunjukan tipe data, bila tidak didefinisikan akan bernilai sesuai dengan data yang diberikan
- Parameter copy berguna untuk menyalin data

Berikut contoh penggunaan data series yang bersumber dari sebuah list python yang bertipe string dan integer.

```
[2]  1  #membuat series tampa index
     2  Sahabat = pd.Series(['Ali', 'Umar', 'Usman', 'Abu Bakar'])
     3  print(Sahabat)
     4
     5  angka = pd.Series([2,4,6,7])
     6  print(angka)
```

```
0          Ali
1         Umar
2        Usman
3    Abu Bakar
dtype: object
0    2
1    4
2    6
3    7
dtype: int64
```

Pada baris ke 1 dan 4 adalah contoh pembuatan series sederhana. Series `Sahabat` merupakan series yang memiliki isi yang bertipe string sedangkan `angka` adalah series yang bertipe integer. Jika kita tidak mendefinisikan indek, maka index akan dibuat secara otomatis muai dari 0 hingga N-1 (dimana N adalah panjang data). Index pada series tidak hanya menggunakan tipe data integer saja tetapi juga dapat tipe data yang lain seperti string. Sebuah indek digunakan untuk mempermudah mengakses series. Berikut ini adalah contoh penggunaan series dengan menggunakan index.

```
[3]  1  #membuat series dengan index
     2  khalifah  = pd.Series(['Ali',  'Umar', 'Usman', 'Abu Bakar'],
     3                        index = ["4th", "2nd", "3rd", "1st"])
     4  print(khalifah )
     5
```

```
4th          Ali
2nd         Umar
3rd        Usman
1st    Abu Bakar
dtype: object
```

Untuk mengakses data, gunakan index seperti "2nd" pada series khalifah dan 0 pada series sahabat.

```
[5]  1  print(khalifah["2nd"])
     2  print(Sahabat[0])
```

```
Umar
Ali
```

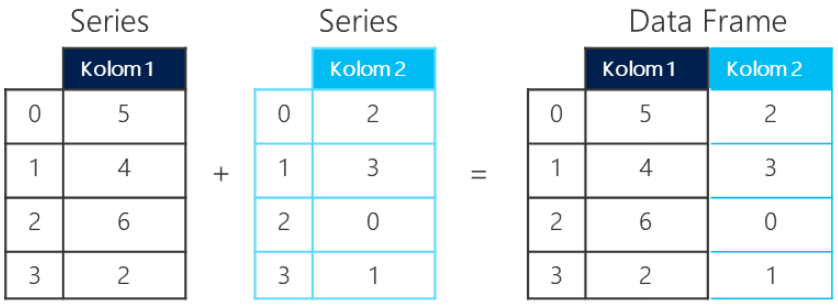
Hasil operasi arimatika tambah pada 2 objek Series mirip dengan operasi join pada pengolahan database. Indeks yang tidak memiliki kesamaan pada 2 objek Series akan memiliki value NaN. Pada contoh berikut ini, kedua series sama-sama memiliki indek "a" dan "b" maka kedua elemen tersebut akan ditambahkan, tetapi untuk index "c" dan "d" bernilai NaN.

```
[6]  1  s1 = pd.Series([1,2,3], index=["a","b","c"])
     2  s2 = pd.Series([4,5,6], index=["a","b","d"])
     3  print(s1+s2)
```

```
a    5.0
b    7.0
c    NaN
d    NaN
dtype: float64
```

- **DataFrame**

Data frame merupakan array dua dimensi dengan baris dan kolom (seperti tabel). Struktur data ini merupakan cara paling standar untuk menyimpan data. Secara sederhana, data frame merupakan tabel atau data tabular, dimana setiap kolom pada Data Frame merupakan objek dari Series, dan baris terdiri dari elemen yang ada pada Series (Gambar 2.3).



Gambar 2.3. Ilustrasi Hubungan *Series* dan *DataFrame*

Untuk membuat sebuah dataframe maka dibutuhkan sebuah *dictionary* data (sebagai kolom) yang terdiri atas nama, umur, warna kesukaan dan tinggi badan. Setiap data pada dictionary akan berisi list yang merepresentasikan data sebagai contoh pada key “nama” akan berisi list ['Ibnu','Asri', 'Barra', 'Aisha'] dan di key “umur” berisi [30, 25, 8, 5]. Setiap elemen yang sama akan menjadi satu baris misalnya nama ibnu akan sebaris dengan umur 30, dan seterusnya. Berikut adalah contoh membuat sebuah dataframe dan perhatikan luarannya dalam bentuk tabel.

```
[7]  1  data = {
     2   'nama': ['Ibnu','Asri', 'Barra', 'Aisha'],
     3   'umur': [30, 25, 8, 5],
     4   'warna_kesukaan': ['merah', 'biru', 'ungu', 'putih'],
     5   'tinggi': [150, 140, 90, 60]
     6   }
     7  df = pd.DataFrame(data)
     8  print(df)
```

```
    nama  umur warna_kesukaan  tinggi
0   Ibnu    30          merah     150
1   Asri    25           biru     140
2  Barra     8           ungu      90
3  Aisha     5          putih      60
```

Ada dua atribut penting pada dataframe yaitu colums dan values. Attribut columns berisi nama kolom yang digunakan dan atribut valus berisi array data yang ada dalam dataframe.

```
[8]  1  df.columns
```

```
Index(['nama', 'umur', 'warna_kesukaan', 'tinggi'], dtype='object')
```

```
[8]  1  #menampilkan data frame
     2  print(df.values)
```

```
[['Ibnu' 30 'merah' 150]
 ['Asri' 25 'biru' 140]
 ['Barra' 8 'ungu' 90]
 ['Aisha' 5 'putih' 60]]
```

Salah satu keunggulan pandas adalah mempermudah dalam pengolahan dan analisis data. Berikut ini beberapa contoh sederhana kemudahan yang ditawarkan oleh pandas:

1. **Pengurutan data**. Untuk mengurutkan cukup memanggil method sort_values lalu memasukkan parameter kolom yang akan menjadi dasar pengurutan. Secara default, pengurutan dilakukan secara menaik (ascending).

```
[9]  1  #pengurutan berdasarkan nama
     2  df.sort_values(by='nama')
```

	nama	umur	warna_kesukaan	tinggi
3	Aisha	5	putih	60
1	Asri	25	biru	140
2	Barra	8	ungu	90
0	Ibnu	30	merah	150

Jika ingin melakukan pengurutan secara menurun tinggal mengubah parameter `ascending` menjadi False. Kemudian jika ingin mengunakan pengurutan pada beberapa kolom maka tinggal mengganti parameter by dengan sebuah list contohnya `by=['nama','umur']` dan parameter arahnya menjadi `ascending=[True,False]` yang berarti pengurutan umur secara menaik dan umur secara menurun.

2. ***Selecting* dan *filtering***. *Selecting* dan *filtering* merupakan kegiatan-kegiatan yang paling banyak dilakukan ketika melakukan persiapan terhadap data seperti *feature selection*, pengecekaan data kosong dan lain-lain. Untuk memilih data pada kolom tertentu cukup dengan mendefinisikan sebuah list kolom dalam braket (tanda "[" dan "]") dataframe.

```
[10]  1  #Menampilkan kolom nama dan tinggi saja
      2  df[['nama','tinggi']]
```

	nama	tinggi
0	Ibnu	150
1	Asri	140
2	Barra	90
3	Aisha	60

Sedangkan untuk pemilihan baris, aturan yang sama seperti pada numpy juga berlaku.

```
[11]  1  #menampilkan 3 baris pertama
      2  df.loc[:2]
```

	nama	umur	warna_kesukaan	tinggi
0	Ibnu	30	merah	150
1	Asri	25	biru	140
2	Barra	8	ungu	90

```
[12]  1  #menampilkan 2 baris terakhir
      2  df.loc[2:]
```

	nama	umur	warna_kesukaan	tinggi
2	Barra	8	ungu	90
3	Aisha	5	putih	60

Sedangkan untuk melakukan *filtering* dengan syarat yang komplek juga mudah dilakukan. Misalnya untuk menampilkan semua item dengan syarat umur lebih besar dari 20 dapat dilakukan dengan df[df[‘umur’]>20].

```
[14]  1  #memfilter dataframe dengasyarat umur>20
      2  df[df['umur']>20]
```

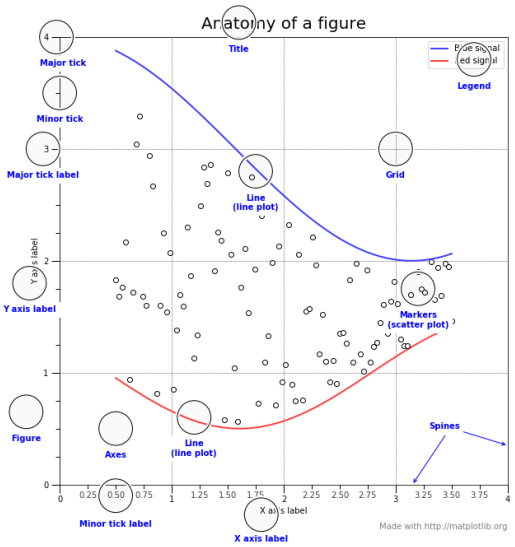
	nama	umur	warna_kesukaan	tinggi
0	Ibnu	30	merah	150
1	Asri	25	biru	140

3. *Loading* dan simpan data. Pandas juga disertai dengan berbagai method yang mempermudah kita untuk membaca atau menyimpan dataset pada aplikasi ML yang kita buat. Berikut contoh sederhana proses penyimpanan data menjadi bentuk CSV, dan file Excel.

```
[14]  1  #menyimpan data kedalam file CSV
      2  df.to_csv('nama_file.csv')
      3  df.to_excel('nama_file.xls')
      4  #membaca/loading data dari csv
      5  df = pd.read_csv('nama_file.csv', sep=',')
```

2.1.5 MatPlotLib

Matplotlib adalah library *plotting* 2D Python yang menghasilkan gambar publikasi bermutu di dalam berbagai format *hardcopy* dan lingkungan interaktif sepanjang platform. Matplotlib dapat digunakan di dalam script Python, shell Python dan ipython, server aplikasi web, dan enam GUI toolkit. Matplotlib mencoba untuk membuat hal mudah menjadi lebih mudah. Kita dapat membuat plot, histogram, power spectra, grafik batang, scatterplot, dll, hanya dengan beberapa baris kode. Gambar 2.4 menunjukkan komponen-komponen dari sebuah chart yang disupport oleh library ini, mulai dari label, lagend, grid dan lain-lain



Gambar 2.4. Anatomi sebuah chart (sumber: matplotlib.org)

Import library

Seperti library python lainnya, setiap penggunaan harus melalui proses import. Caranya sama seperti proses import pada Numpy atau Panda.

```
[1]  1 import matplotlib as mpl
     2 import matplotlib.pyplot as plt
     3 %matplotlib inline
     4
     5 plt.style.use('seaborn-pastel')
```

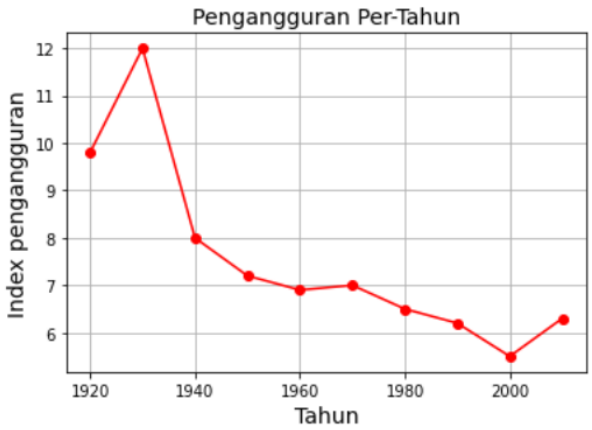
Agar dapat menggunakan library ini pada Ipython Notebook maka perintah pada baris ke-3, %matplotlib, diperlukan. Pada IPython notebook, grafis yang dihasilkan akan langsung ditampilkan pada halaman notebook. Sedangkan baris ke-5 untuk menentukan style apa yang akan digunakan dalam chart dalam yang dihasilkan. Berikut ini beberapa contoh penggunaan matplotlib:

Line Chart

Line Chart atau *line plot* atau *line graph* atau *curve chart* adalah sebuah tipe gradis yang digunakan untuk memampilkan sejumlah data titik yang biasa disebut "marker", lalu marker tersebut dihubungkan dengan garis lurus. Grafis ini terdiri atas dua sumbu yaitu sumbu tegak (vertikal) dan sumbu mendatar (horizontal). Biasanya jenis grafis ini digunakan untuk memvisualisasikan sebuah trend pada data pada interval waktu tertentu. Biasanya sumbu mendatar digunakan untuk menunjukkan waktu.

Perhatikan contoh berikut [3], diberikan dua buah list data yaitu data tahun dari 1920 sampai 2010 dan data pengangguran (variabel `Tahun` dan `Pengangguran` pada baris ke-1 dan ke-2). Untuk membuat plot maka cukup memanggil method `plt.plot`. Method tersebut membutuhkan dua parameter wajib sebagai x dan y serta beberapa parameter opsional seperti warna dan jenis marker. Pada contoh berikut digunakan empat parameter `x, y, color` dan `marker`. Pada baris ke-4 digunakan untuk membuat judul chart dan baris ke-5 dan ke-6 untuk membuat label pada chart.

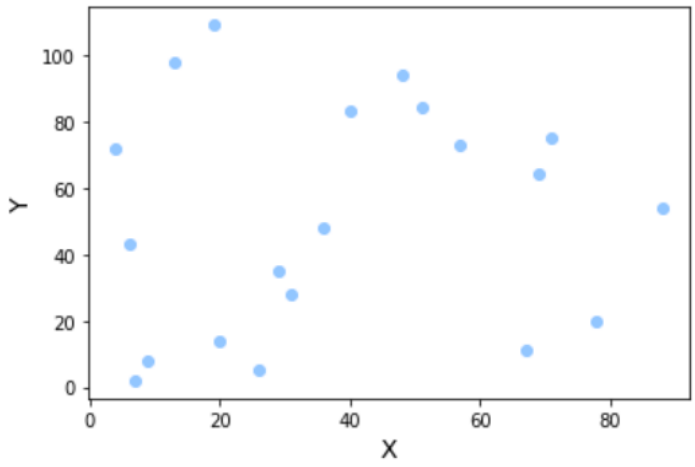
```
[3]  1 Tahun = [1920,1930,1940,1950,1960,1970,1980,1990,2000,2010]
     2 Pengangguran = [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3]
     3 plt.plot(Tahun, Pengangguran, color='red', marker='o')
     4 plt.title('Pengangguran Per-Tahun', fontsize=14)
     5 plt.xlabel('Tahun', fontsize=14)
     6 plt.ylabel('Index pengangguran', fontsize=14)
     7 plt.grid(True)
     8 plt.show()
```



Plot Sebaran (*Scatter Plot*)

Sebuah plot sebaran adalah sebuah grafik yang menunjukkan hubungan antara dua set data, seperti hubungan antara umur dan tinggi. Plot ini mirip dengan *line plot*, tetapi dalam *plot* ini setiap *marker* tidak dihubungkan dengan garis. Kali ini kita akan menggambar sebuah plot sebaran menggunakan matplotlib. Sebagai contoh buat dua set data, x dan y (pada baris ke-1 dan ke-2), lalu panggil method scatter pada baris ke 3 menggunakan dua parameter yaitu x dan y untuk membuat grafisnya. Baris ke-4 dan ke-5 untuk menampilkan nama label pada grafis:

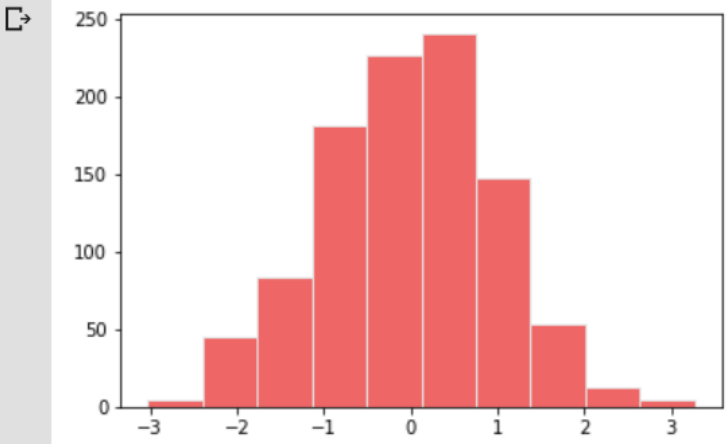
```
[4]  1 x = [20,4,6,7,9,13,19,26,29,31,36,40,48,51,57,67,69,71,78,88]
     2 y = [14,72,43,2,8,98,109,5,35,28,48,83,94,84,73,11,64,75,20,54]
     3 plt.scatter(x,y)
     4 plt.xlabel('X', fontsize=14)
     5 plt.ylabel('Y', fontsize=14)
     6 plt.show()
     7
```



Histogram

Kata histogram berasal dari bahasa Yunani: histos, dan gramma. Pertama kali digunakan oleh Karl Pearson pada tahun 1895 untuk memetakan distribusi frekuensi dengan luasan area grafis batangan menunjukkan proporsi banyak frekuensi yang terjadi pada tiap kategori. Sebuah histogram adalah grafik yang menampilkan frekuensi data menggunakan batang, dimana angka dikelompokkan dalam rentang atau kelompok tertentu. Dengan kata lain, frekuensi setiap elemen data di dalam daftar ditunjukkan menggunakan histogram. Angka yang dikelompokkan dalam bentuk rentang tertentu disebut bins (kelompok):

```
[5]  1 # mencoba mengubah warna chart dan edgenya
     2 x = np.random.randn(1000)
     3 plt.hist(x, edgecolor='#E6E6E6', color='#EE6666');
```



Pada contoh, sebuah dataset x merupakan 1000 nilai acak, lalu dengan menggunakan method `plt.hist` untuk membuat sebuah histogram dengan warna `#EE6666`.

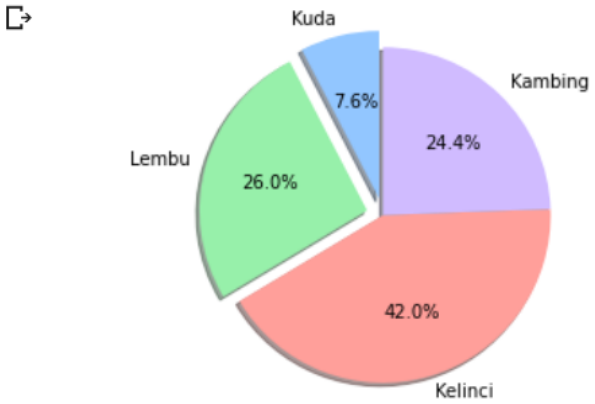
Pie Chart

Pie chart atau diagram lingkaran merupakan grafik yang berbentuk lingkaran yang dibagi menjadi beberapa irisan dan luasnya bergantung kepada proporsi numerik atau kuantitas dari data yang dimiliki. Satu lingkaran menunjukkan bagian utuh atau seratus persen. Setengah lingkaran menunjukkan proporsi setengah dari total atau lima puluh persennya, dan seterusnya.

Pada contoh [6], ditunjukkan sebuah pie chart hewan ternak pada sebuah peternakan dimana pichart tersebut terdiri atas empat label yang mewakili hewan ternak yaitu Kuda, Lembu, Kelinci dan Kambing. Selanjutny ada variabel sizes yang menunjukkan jumlah hewan-hewan tersebut sesuai urutan. Lalu library akan otomatis mengkonversi jumlah-jumlah hewan tersebut kedalam persentase dimana 10 ekor kuda mewakili 7.6% dari total area chart, 34 ekor lembu mewakili 26% dari total area chart, 55 ekor kelinci

mewakili 42% dari total area chart, dan 32 ekor kambing mewakili 24.6% dari total area chart.

```
[6]  1 # Memampilkan Pie chart
     2 labels = 'Kuda', 'Lembu', 'Kelinci', 'Kambing'
     3 sizes = [10, 34, 55, 32] #ukuran Pie chart
     4 posisi = (0.1, 0.1, 0, 0)  # elemen 1 & 2 akan menjorok keluar
     5 fig1, ax1 = plt.subplots()
     6 ax1.pie(sizes, explode=posisi, labels=labels, autopct='%1.1f%%',
     7           shadow=True, startangle=90)
     8 ax1.axis('equal')
     9 plt.show()
```



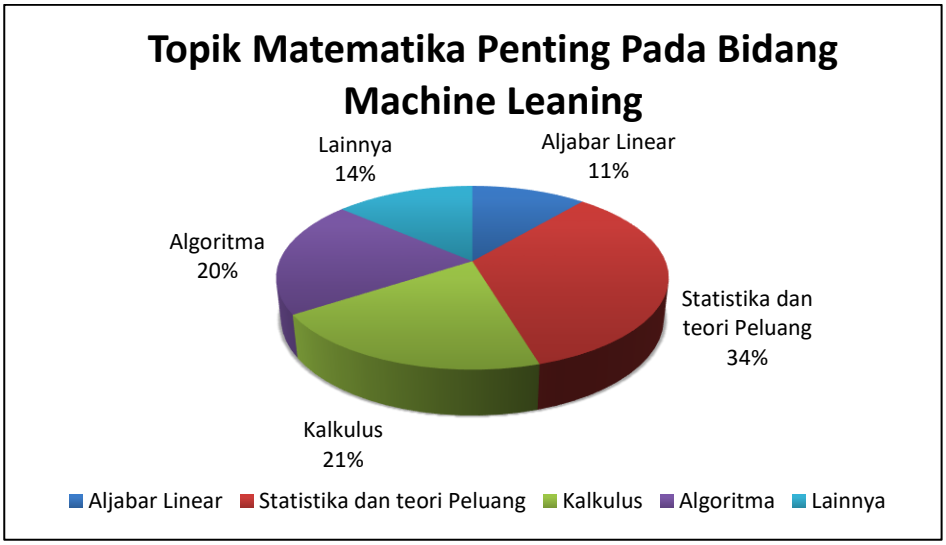
2.2 Pondasi Matematika dan Statistika

Kemampuan matematika hukumnya adalah **WAJIB** jika anda ingin serius dibidang ini. Adapun kemampuan yang harus ada sebelum anda mempelajari ML adalah Aljabar Linier, Kalkulus, Probabilitas dan Statistik. Pada buku ini tidak dibahas secara mendalam tentang pondasi-pondasi matematika yang anda butuhkan tetapi akan disebut topik-topik yang perlu anda kuasai ketika belajar ML. Adapun topik-topik matematika yang harus digaris bawahi adalah

- **Aljabar linier** adalah sub-bidang matematika yang berkaitan dengan vektor, matriks, dan transformasi linier. Jika anda berasal dari jurusan ilmu komputer, materi ini bukanlah materi yang asing. Menurut penulis, ini adalah **Pondasi Kunci** untuk bidang ML, dari notasi yang digunakan untuk menggambarkan operasi algoritma hingga implementasi algoritma dalam kode. Aljabar linier ini muncul hampir di semua topik machine learning seperti Operasi Matrix , *Eigenvalues & Eigenvectors*, *Vector Spaces*, Regresi Linier, *Eigen Decomposition*, *Singular Value Decomposition* (SVD), *Principal Component Analysis* (PCA) dan lain-lain.

- **Teori peluang dan statistika** memberikan kontribusi paling besar dalam bidang ini. Apapun materi-materi Teori peluang dan statistika yang diperlukan ML adalah Kombinatorika, Teorema bayes, Aturan Probabilitas, Kondisional dan *Joint Distributions*, *Standard Distributions* (Bernoulli, Binomial, Multinomial, Uniform dan Gaussian), *Maximum Likelihood Estimation* (MLE), dan Metode Sampling.

- **Kalkulus** beberapa materi yang harus dipelajari adalah Turunan dan Integral, Fungsi Vector-Values, *Directional Gradient*, Hessian, Jacobian, Laplacian dan Lagragian Distribution.



Gambar 2.5. Topik-topik Matematika yang Penting di ML (sumber: Wale Akinfaderin, 2017)

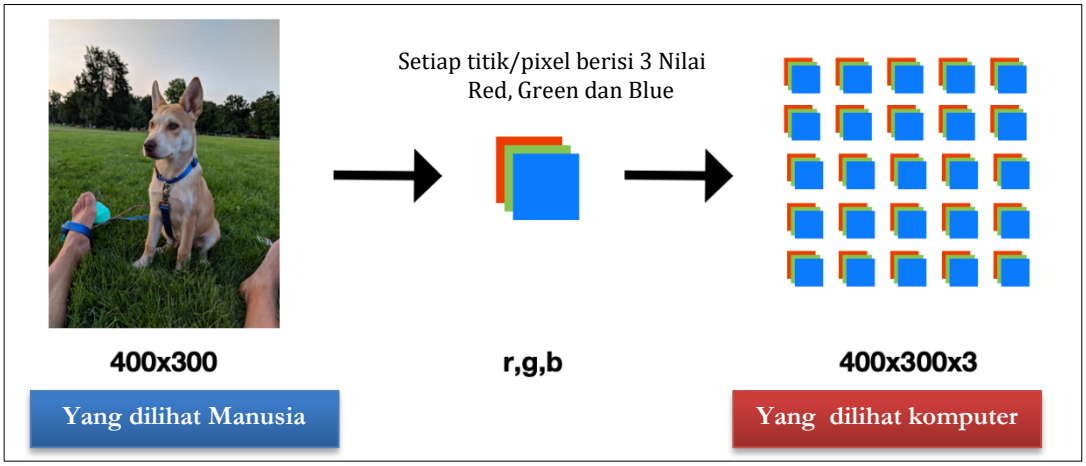
- **Teori matermatika lainnya** adalah topik-topik matematika yang dibutuhkan tetapi tidak masuk kedalam kategori lainnya Analisa Real dan Complex (Sets dan Sequences, Topology, Metric Spaces, Limits, *Cauchy Kernel*, *Fourier Transforms*), Teori Informasi(Entropy, *Information Gain*), *Function Spaces* dan Manifolds.

Berikut ini akan dibahas beberapa topik yang penulis anggap penting untuk diingat kembali sebagai dasar machine learning

2.2.1 Aljabar Linier

Aljabar linier adalah cabang dari matematika yang menjadi syarat utama bagi orang-orang yang belajar ML. Melalui Aljabar Linier-lah banyak teori dan metode ML ditemukan dan dapat digunakan untuk memecahkan masalah terkait data. Pengetahuan yang kokoh tentang Aljabar Linier akan membuat perjalanan ML anda lebih fokus. Sederhananya, aljabar linier ini dapat dikatakan matematikanya data. Jika kita ingin merepresentasikan data maka bahasa yang akan kita pilih adalah scalar, vektor, matrik atau tensor. Dengan kata lain, ketika kita berusaha mengolah data maka sebenarnya kita mengolah scalar, vektor, matrik ATAU tensor.

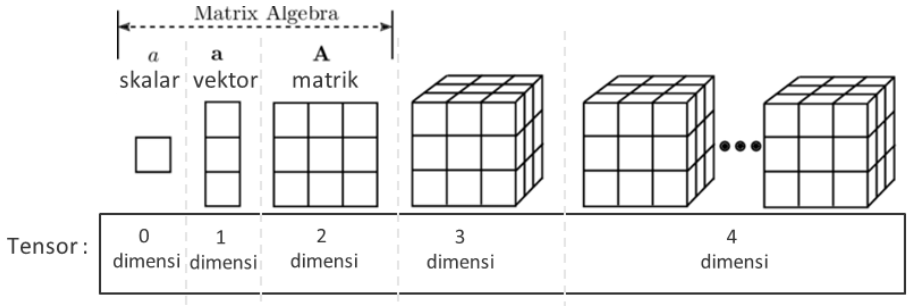
Sebagai contoh perhatikan gambar anjing pada Gambar 2.6. Sebagai manusia, kita akan melihat kaki manusia dan seekor anjing coklat yang duduk di atas rumput. Tetapi bagi komputer, sebuah gambar tidak lebih hanya sekumpulan angka. Setiap titik digambar yang disebut piksel yang merupana satuan terkecil dari sebuah gambar. Sebuah piksel terdiri atas tiga nilai yaitu *Red, Green* dan *Blue*. Secara matematis, sebuah piksel dapat dikatakan sebuah vektor karena berisi tiga nilai dan sebuah gambar terdiri atas susunan pixel-pixel dalam bentuk matrik sehingga membentuk sebuah tensor.



Gambar 2.6. Bagaimana Manusia & Komputer Melihat Gambar

Pengolahan terhadap gambar seperti memutar (*rotasi*) atau memperbesar atau memperkecil, menggabungkan dan lain-lain merupakan proses pengolahan scalar, vektor, matrik atau tensor. Materi tentang pengolahan-pengolahan tersebut merupakan sub bagian dari ilmu aljabar linear.

Berikut ini penjelasan rinci mengenai Scalar, Vektor, Matrik dan Tensor serta operasi-operasi dasar yang dapat dilakukan pada masing-masing objek:



Gambar 2.7. Perbedaan Scalar, Vektor, Matrik dan Tensor

Skalar

Skalar merupakan sebuah angka. Biasanya dilambangkan dengan huruf kecil ditulis miring seperti "*x*". Tipe data sebuah skalar tidak hanya integer tetapi juga dapat berisi *natural number* (3, 4, 5, …) , *rational number* (4/3, 1.666666…), atau *irrational number* .

Vektor

Vektor merupakan sebuah elemen yang mendasar dalam aljabar linier. Vektor adalah sekumpulan angka yang terurut (*an ordered finite list*). Vektor sering digunakan pada ML untuk mendeskripsikan algoritma dan proses-proses seperti mendefinisikan variabel target (y) pada proses pelatihan algoritma. Sebuah vektor dibangun oleh komponen-komponen yang biasanya berupa angka. Dapat dianalogikan bahwa vektor itu adalah sekumpulan angka. Notasi yang digunakan untuk vektor biasanya direpresentasikan dengan **huruf kecil**, contoh

$$\mathrm{v} = (v_1, v_2, v_3) \text{ atau } \mathrm{v} = (21, 52, 33)$$

Sebagian buku juga sering menotasikannya secara horizontal

$$\mathrm{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \text{ atau } \mathrm{v} = \begin{pmatrix} 21 \\ 52 \\ 33 \end{pmatrix}$$

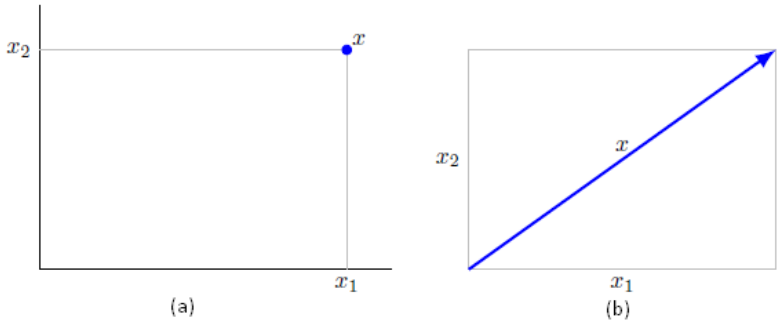
Elemen (isi atau komponen) dari sebuah vektor adalah nilai-nilai dari array v, sedangkan ukuran (dimensi) dari vektor adalah jumlah semua elemen (*count*, bukan sum nilai) yang ada didalam vektor. Sebagai contoh

vektor v memiliki ukuran = 3, dan elemen ke-2 adalah 52. Sebuah vektor dengan ukuran n biasanya disebut n-vector.

Berikut ini beberapa contoh penggunaan vektor dalam kehidupan sehari-hari

- **Lokasi dan perpindahan**. Sebuah vektor dengan ukuran 2 (*2-vector*) dapat digunakan untuk merepresentasikan sebuah titik atau lokasi dalam sebuah ruang dua dimensi seperti pada Gambar 2.8. Selanjutnya matrik berukuran 3 (*3-vector*) dapat juga merepresentasikan titik pada ruang tiga dimensi.

  Selain itu, ia juga dapat merepresentasikan perpindahan pada ruang dua dimensi ataupun 3 dimensi. Biasanya perpindahan ini dilambangkan dengan sebuah panah. Sebuah vektor juga dapat merepresentasikan kecepatan atau percepatan, pada waktu tertentu, dari suatu titik yang bergerak di ruang 2-D atau 3-D.



Gambar 2.8. Vektor berukuran 2 menggambarkan posisi titik x dengan koordianat x_1,x_2 (gambar a) dan sebuah vektor x yang merepresentasikan perpindahan sejauh x_1 pada sumbu x dan x_2 pada sumbu y (gambar b)

- **Warna**. Sebuah vektor berukuran 3 dapat digunakan untuk menyimpan nilai sebuah warna yang terdiri atas tiga nilai dari warna merah (*Red*), hijau (*Green*) dan biru (*Blue*). Misalnya vektor (0,0,0) merepresentasikan warna hitam, vektor (255,0,0) merepresentasikan warna merah, vektor (0,255,0) merepresentasikan warna hijau, vektor (0,0,255) merepresentasikan warna merah dan vektor (255,255,0)

merepresentasikan warna kuning yang merupakan campuran dari warna merah dan hijau.

- **Portofolio Saham**. Sebuah vektor s berukuran n dapat mewakili portofolio saham atau investasi dalam n aset yang berbeda, dimana s_i merupakan jumlah saham atau aset yang dimiliki oleh i. Sebagai contoh, Vektor (100; 50; 20) mewakili portofolio yang terdiri dari 100 saham milik orang ke-1, 50 saham milik orang ke-2, dan 20 saham milik orang ke-3.
- **Data Pembelian**. Sebuah vektor p berukuran n dapat mewakili sejumlah data pembelian yang dilakukan pada sebuah periode waktu tertentu. p_i merepresentasikan data pembelian ke i.

Untuk merepresentasikan sebuah vektor pada menggunakan Python dapat memanfaatkan NumPy. Sebagai contoh kita akan membuat sebuah vektor dengan nilai 1, 2 dan 3 adapun fungsi np.array adalah fungsi yang digunakan untuk membuat sebuah vektor.

```
[1]  1 import numpy as np
     2 # Membuat sebuah vektor
     3 a = np.array([1, 2, 3])
     4 print(a)
     5
```

```
[1 2 3]
```

Adapun operasi-operasi vektor yang harus kita ketahui adalah

1. Operasi Penambahan Vektor

Dua buah vektor yang panjang atau ukurannya sama dapat ditambahkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama. Pada penjumlahan ini berlaku sifat komunitatif dan asosiatif .

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} + \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = \begin{pmatrix} v_1 + i_1 \\ v_2 + i_2 \\ v_3 + i_3 \end{pmatrix}$$

Implementasinya dalam bahasa Python sangat sederhana.

```
[2]  1 # Membuat sebuah array
     2 a = np.array([1, 2, 3])
     3 b = np.array([4, 5, 6])
     4 # penambahan vector
     5 c = a + b
     6 print(a,"+",b,"=",c)
     7
```

```
[1 2 3] + [4 5 6] = [5 7 9]
```

Penjelasannya hasil dapat dilihat pada contoh matriks berikut

$$v = \begin{pmatrix}1\\2\\3\end{pmatrix} + \begin{pmatrix}4\\5\\6\end{pmatrix} = \begin{pmatrix}1+4\\2+5\\3+6\end{pmatrix} = \begin{pmatrix}5\\7\\9\end{pmatrix}$$

2. Operasi Pengurangan Vektor

Sama seperti penambahan, dua buah vektor yang panjang atau ukurannya sama dapat dikurangkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama.

$$v = \begin{pmatrix}v_1\\v_2\\v_3\end{pmatrix} - \begin{pmatrix}i_1\\i_2\\i_3\end{pmatrix} = \begin{pmatrix}v_1 - i_1\\v_2 - i_2\\v_3 - i_3\end{pmatrix}$$

```
[3]  1 # Membuat sebuah array
     2 a = np.array([1, 2, 3])
     3 b = np.array([4, 5, 6])
     4 # penambahan vector
     5 c = b - a
     6 print(b," - ",a,"=",c)
     7
```

```
[4 5 6]  -  [1 2 3] = [3 3 3]
```

3. Operasi Perkalian Vektor

Sama seperti penambahan, dua buah vektor yang panjang atau ukurannya sama dapat dikurangkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama.

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \mathrm{X} \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = \begin{pmatrix} v_1 \mathrm{\ x\ } i_1 \\ v_2 \mathrm{\ x\ } i_2 \\ v_3 \mathrm{\ x\ } i_3 \end{pmatrix}$$

```
[4]  1 a = np.array([1, 2, 3])
     2 b = np.array([4, 5, 6])
     3 # perkalian vector
     4 c = b * a
     5 print(b," x ",a,"=",c)
```

```
[4 5 6]  x  [1 2 3] = [ 4 10 18]
```

4. Operasi Pembagian Vektor

Sama seperti penambahan, dua buah vektor yang panjang atau ukurannya sama dapat dikurangkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama.

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} / \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = \begin{pmatrix} v_1 \,/\, i_1 \\ v_2 \,/\, i_2 \\ v_3 \,/\, i_3 \end{pmatrix}$$

```
[5]  1 # Membuat sebuah array
     2 a = np.array([8, 8, 8])
     3 b = np.array([1, 2, 3])
     4 # penambahan vector
     5 c = b / a
     6 print(b," / ",a,"=",c)
     7
```

```
[1 2 3]  /  [8 8 8] = [0.125 0.25  0.375]
```

5. Operasi Dot Product

Sama seperti penambahan, dua buah vektor yang panjang atau ukurannya sama dapat dikurangkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama.

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} . \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = (v_1 i_1 + v_2 i_2 + v_3 i_3)$$

```
[6]  1 # Membuat sebuah array
     2 a = np.array([2, 2, 2])
     3 b = np.array([1, 2, 3])
     4 # penambahan vector
     5 c = np.dot(a,b)
     6 print(b," / ",a,"=",c)
     7
```

```
[1 2 3]  /  [2 2 2] = 12
```

Matrik

Matrik adalah susunan persegi panjang dari bilangan-bilangan yang diatur dalam baris dan kolom. Adapun attribut yang penting adalah dimensi dari matrik yang menunjukkan jumlah baris dan kolom. Notasi yang digunakan untuk merepresentasikan matrik adalah

$$v = ((v_{11}, v_{12})(v_{21}, v_{22}), (v_{31}, v_{32}))$$

Sebagian buku juga sering menotasikannya matrik secara horizontal

$$v = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \\ v_{31} & v_{32} \end{pmatrix}$$

Untuk merepresentasikan sebuah vektor pada menggunakan Python dapat memanfaaykan NumPy. Sebagai contoh kita akan membuat sebuah vektor dengan nilai 1, 2 dan 3 adapun fungsi np.array adalah fungsi yang digunakan untuk membuat sebua vektor..

```
[7]  1 # Membuat sebuah matrix
     2 a = np.array ([[1, 2, 3], [4, 5, 6]])
     3 print(a)
```

```
[[1 2 3]
 [4 5 6]]
```

Operasi-operasi yang terdapat dimatrix relatif sama dengan fungsi yang ada di vektor. Berikut ini beberapa operasi yang sering dilakukan pada matrik menggunakan numpy

Transpose

```
[8]  1 #mentranspose matrix a
     2 a = np.array ([[1, 2, 3], [4, 5, 6]])
     3 b = np.transpose(a)
     4 print(b)
```

```
[[1 4]
 [2 5]
 [3 6]]
```

Inverse

```
[9]  1 #menginverse matrix X
     2 x = np.array([[1,2],[3,4]])
     3 y = np.linalg.inv(x)
     4 print (y)
```

```
[[-2.   1. ]
 [ 1.5 -0.5]]
```

2.2.2 Statistika

Statistika adalah sebuah ilmu yang mempelajari bagaimana cara merencanakan, mengumpulkan, menganalisis, lalu menginterpretasikan, dan akhirnya mempresentasikan data. Singkatnya, statistika adalah ilmu yang bersangkutan dengan suatu data. Ilmu ini dapat dikatakan adalah pondasi dasar dari Machine Learning, oleh karena itu, pembaca harus dapat memahami, setidaknya konsep-konsep dasar dalam statistika. Berikut konsep-konsep yang banyak muncul dalam ML.

2.2.2.1 PROBABILITAS

Probabilitas atau sering disebut peluang sering diasosiasikan dengan paling tidak sebuah kejadian. Kejadian ini bisa apa saja, sebagai contoh pelemparan dadu, pengambilan bola berwarna dan lain-lain. Pada contoh tersebut semua luaran yang dihasilkan akan bersifat acak (random), misalnya kecepatan anda mengocok dadu tidak akan mempengaruhi hasil luarannya. Biasanya variabel yang merepresentasikan luaran kejadian disebut variabel acak (*random variable*).

Ada banyak pengertian probabilitas, salah satunya dari perspektif statistik adalah rasio frekuensi suatu item atau kejadian. Berikut beberapa contoh probabilitas:

1. Sebuah mata uang logam mempunyai sisi dua (Huruf dan Gambar), Jika uang tersebut dilambungkan satu kali, maka peluang untuk keluar sisi Huruf adalah ½.

2. Sebuah dadu yang memiliki enam sisi dilemparkan maka peluang untuk keluar mata 'lima' saat pelemparan dadu tersebut satu kali adalah 1/6 (karena banyaknya permukaan dadu adalah 6)

Dalam suatu percobaan semua hasil yang mungkin disebut ruang sampel, sedangkan setiap anggota dalam ruang sampel disebut titik sampel dan hasil yang diharapkan disebut kejadian. Peluang kejadian A dinotasikan

dengan P(A) adalah perbandingan banyaknya hasil kejadian A dinotasikan n(A) terhadap banyaknya semua hasil yang mungkin dinotasikan dengan n(S) dalam suatu percobaan.

Secara Empiris maka dapat dikatakan persamaan peluang adalah:

$$p(E) = \frac{X}{N} \quad (2\text{-}1)$$

p: Probabilitas
E: Event (Kejadian)
X: Jumlah kejadian yang diinginkan (peristiwa)
N: Keseluruhan kejadian yang mungkin terjadi

Kisaran nilai peluang suatu kejadian A adalah $0 \leq P(A) \leq 1$. Jika $P(A) = 0$ disebut kemustahilan dan $P(A) = 1$ disebut kepastian.

DEFINISI

Ruang sampel adalah himpunan dari semua hasil yang mungkin pada suatu percobaan/kejadian. Contoh: Pada pelemparan sebuah dadu, maka ruang sampelnya adalah S = {1,2,3,4,5,6}

Titik sampel adalah anggota-anggota dari ruang sampel atau kemungkinan-kemungkinan yang muncul. Contoh: Pada pelemparan sebuah dadu, maka titik sampelnya : (1), (2), (3), (4), (5), dan (6)

Selain itu, probabilitas juga merupakan sebuah *tool* yang dapat kita gunakan untuk mengukur tingkat kepercayaan (*degree of belief*), biasanya disebut Probabilitas Bayesian. Teori probabilitas Bayesian merupakan satu dari cabang teori statistik matematik yang memungkinkan kita untuk membuat satu model ketidakpastian dari suatu kejadian yang terjadi dengan menggabungkan pengetahuan umum dengan fakta dari hasil pengamatan.

2.2.2.2 CENTRAL TENDENCY

Teori *central tendency*, mungkin teori yang paling tua yang pernah kita kenal. Sejak duduk dibangku sekolah menengah kita telah diperkenalkan dengan teori-teori ini. Sebuah pengukuran *central tendency* melambangkan sebuah nilai yang mencoba mendeskripsikan dataset dengan mengidentifikasi nilai "posisi tengah" dataset. Pengukuran *central tendency* adalah sebagai berikut

1. Mean.

   Biasanya disebut "rata-rata". Pengukuran ini dapat digunakan pada data *continue*. Formula dari rata-rata adalah jumlah dari seluruh item dibagi dengan jumlah item. Pengukuran ini menggambarkan nilai tengah dari populasi yang diamati. Adapun formula mean adalah

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad (2\text{-}2)$$

   Rata-rata tidak mempertimbangkan pencaran (variabilitas) nilai, sehingga sebelum melakukan interpretasi atas nilai rata-rata perlu melihat variabilitasnya. Perhatikan contoh data umur mahasiswa berikut:

MAHASISWA	1	2	3	4	5	6	7	8	9	10
UMUR	19	18	20	21	18	19	24	23	50	55

   Berdasarkan pengukuran, maka rata-rata umur mahasiswa adalah 26.7 tahun. Namun, jika diperhatikan pada data, kebanyakan mahasiswa berada pada umur 18 sampai 24 tahun. Namun karena ada dua mahasiswa yang merupakan pencilan sehingga jika hanya memperhatikan rata-rata maka kita akan kurang tepat dalam menginterpretasikan data. Oleh karena itu, diperlukan pengukuran lainnya seperti median, mode, kurtosis dan lain-lain untuk menginterpretasikan data dengan tepat.

2. Median

   Median adalah nilai tengah dari dataset terurut. Pada median, nilainya tidak terpengaruh kepada outlier. Adapun formula untuk menemukan median adalah

$$Median(X) = \begin{cases} X\left[\frac{n}{2}\right] & Jika\ n\ bernilai\ genap \\ X\left[\frac{n-1}{2}\right] + X\left[\frac{n+1}{2}\right] \Big/ 2 & Jika\ n\ bernilai\ ganjil \end{cases} \quad (2\text{-}3)$$

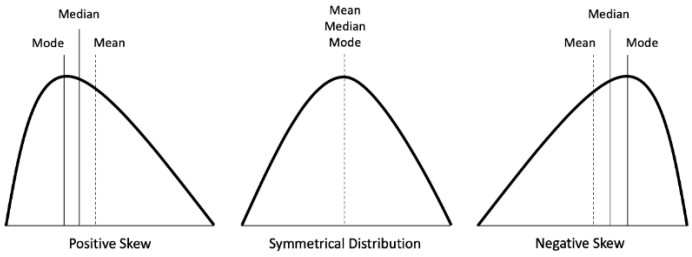
   Jika jumlah data kita genap, maka nilai median kita berada di antara kedua nilai yang berada di tengah; sebagai contoh jika kita mempunyai data 1,1,2,3,3,4,4,5 maka nilai median kita adalah 3 karena kita mempunyai 8 data sehingga nilai titik tengahnya berada di posisi ke-4 (nilai 3) dan ke-5 (nilai 3) dan lebih tepatnya lagi data di kedua posisi ini ditambahkan lalu dibagi 2.

   Median lebih sering digunakan jika Mean tidak mampu menjelaskan data kita dengan baik, sehingga diperlukan pengukuran titik tengah menggunakan pengukuran lain.

3. Mode

   Mode atau modus adalah nilai yang paling sering muncul dalam suatu dataset. Modus juga merupakan nilai mayoritas atau nilai dengan frekuensi paling tinggi. Perhitungan modus dapat diterapkan pada data numerik maupun data kategoris, contohnya dalam menentukan data dari warna paling banyak disukai siswa dan mayoritas nilai ulangan sebuah mata pelajaran yang diperoleh oleh siswa dalam suatu kelas.

Dalam menganalisa data menggunakan prinsip *central tendency* biasanya kita akan bertemu istilah Skewness dan Kurtosis. Istilah 'skewness' digunakan untuk berarti tidak adanya simetri dari rata-rata dataset. Perhatikan gambar 2.9



Gambar 2.9. *Skewness* pada data

Data dapat didistribusikan dengan banyak cara, seperti menyebar lebih banyak di sebelah kiri atau di sebelah kanan atau merata. Ketika data tersebar secara seragam di titik pusat, itu disebut Distribusi Normal. Kurva ini sangat simetris, berbentuk lonceng, misalnya kedua sisinya sama, sehingga tidak miring. Pada kondisi ini disebut distribusi simetris (*symmetrical distribusion*) jika mean, median, dan mode terletak pada satu titik. Jika mean lebih kecil dari median atau mode maka distribusi akan menunjukkan kemiringan negatif, begitu juga sebaliknya jika mean lebih besar dari mode atau median maka terjadi kemiringan positif.

Dalam statistik, kurtosis didefinisikan sebagai parameter ketajaman relatif dari puncak kurva distribusi probabilitas. Ini memastikan cara pengamatan dikelompokkan di sekitar pusat distribusi. Ini digunakan untuk menunjukkan kerataan atau puncak dari kurva distribusi frekuensi dan mengukur ekor atau *outlier* dari distribusi.

Kurtosis positif menyatakan bahwa distribusi lebih memuncak daripada distribusi normal, sedangkan kurtosis negatif menunjukkan bahwa distribusi kurang memuncak daripada distribusi normal. Ada tiga jenis distribusi:

- Leptokurtik: Tajam memuncak dengan ekor gemuk, dan kurang bervariasi.
- Mesokurtik: Sedang memuncak
- Platykurtic: Puncak paling rata dan sangat tersebar

2.2.2.3 VARIABILITAS

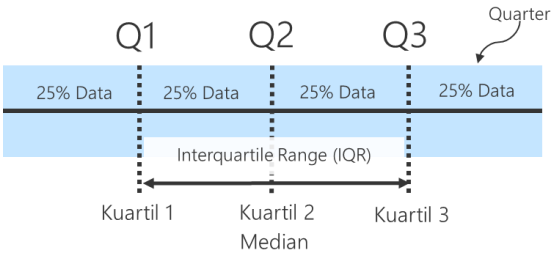
Variabilitas (ukuran penyebaran) merupakan suatu nilai dari dataset yang menjadi ukuran untuk mengetahui besarnya penyimpangan data dengan nilai rata-rata hitungnya atau menggambarkan bagaimana suatu dataset yang menyebar terhadap pusatnya data.

Variabilitas berfungsi sebagai alat ukur untuk mengetahui homogenitas dan heterogenitas suatu kelompok data. Semakin kecil nilai variabilitas menunjukkan semakin kecil penyebaran suatu kelompok data (homogenitas). Semakin besar nilai variabilitas menunjukkan semakin besar penyebaran suatu kelompok data (heterogenitas).

Berikut ini beberapa konsep dalam variabilitas yang penting untuk diketahui diantaranya

- *Range* adalah perbedaan antara data numerik terkecil hingga data terbesar. Tidak banyak informasi yang bisa kita ketahui selain mengetahui perbedaan besar data kita
- *Quartile* adalah nilai-nilai yang membagi data menjadi 4 bagian. Bagian yang terbagi disebut dengan *Quarter*. Jika kita menyebut *Quartile*, maka yang sebenarnya kita acu adalah nilai yang membagi, bukan hasil bagiannya.

  Terdapat 3 jenis *Quartile* pada saat menggunakan metode Quartile, yaitu Q1 (Nilai antara median dengan data terkecil), Q2 (Median), dan Q3 (Nilai antara median dengan terbesar).



Gambar 2.10. Quartile membagi data yang diurutkan menjadi 4 bagian. Setiap bagian yang terbagi disebut dengan quarter.

- *Variance* adalah pengukuran suatu variabilitas dari data untuk mengetahui seberapa jauh data yang dimiliki tersebar.

  Variance dihitung berdasarkan total dari setiap data (x_i) dikurangi dengan mean data ($\bar{x}$). Sedikit perbedaan jika kita berbicara mengenai data pada populasi dan sampel. Pada *variance* populasi kita membagi data kita dengan seluruh jumlah sampel data (N), sedangkan jika data sampel maka kita membaginya dengan jumlah data yang ada dikurangi 1 (N-1). Ini dilakukan karena data sampel memiliki ketidakpastian dibandingkan populasi sehingga kita memperbesar perhitungan persebaran kita.

  Menggunakan *variance* berarti kita menjelaskan dasar kita melalui titik tengah *mean*, sehingga *variance* dapat menjelaskan seberapa tersebar data kita dari *mean* dan satu sama lainnya. Jika *variance* kita kecil, maka data kita tersebar dekat dengan nilai *mean* sedangkan jika nilai *variance* besar menandakan data kita semakin tersebar jauh dari *mean* dan dengan satu sama lain. Selain itu, karena *variance* melakukan pemangkatan dari data maka data yang semakin jauh nilainya akan semakin dibesarkan sehingga kita bisa memperkirakan seberapa banyak data yang jauh dari *mean*.

- *Standard Deviation*. *Standard deviation* adalah *measure of spread* yang paling sering digunakan karena memberikan informasi yang jelas dan intuitif. Untuk mendapatkan nilai Standard deviation kita hanya perlu melakukan akar kuadrat terhadap *variance*.

  Standard deviation menggambarkan seberapa berbeda nilai di data kita terhadap mean. Jika menggunakan bahasa sehari-hari, standard deviation adalah nilai plus-minus dari mean ($\bar{x} \pm s$). Selain itu standar deviation juga digunakan di dalam empirical rule atau 68–95–99.7 rule dimana normalnya data kita tersebar sebesar ± 1 * STD (68 % data), ± 2 * STD (95 % data), dan ± 3 * STD (99.7% data). Data yang terletak lebih atau kurang dari batas tersebut menandakan bahwa data tersebut adalah suatu outlier.

2.2.2.4 PROBABILITY DISTRIBUTION

Probability Distribution adalah suatu fungsi matematika yang memberikan probabilitas (*likelihood*) dari variabel acak untuk memiliki suatu nilai. Dengan kata lain, kemungkinan suatu variabel acak untuk memiliki suatu nilai akan tergantung dari *Probability Distribution* .

Bab

3

Machine Learning Lifecycle

Dan mintalah pertolongan dengan sabar dan shalat. – (Q.S Al-Baqarah: 45)

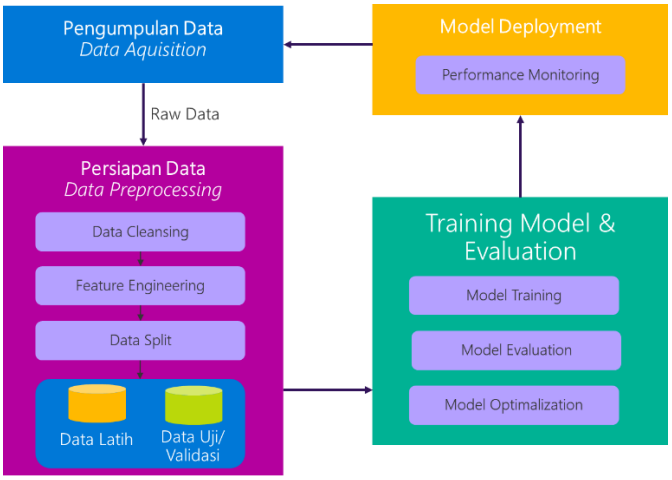
Capaian Pembelajaran:

- Mampu mengerahui dan memahami Machine Learning Lifecyle
- Memahami konsep data dan jenis-jenis dataset pada ML
- Mampu mengidentifikasi dan mengumpulkan data sederhana
- Mampu melakukan *data cleansing*, *feature extraction* dan *data split*
- Mampu mempersiapkan data training dan data testing untuk pelatihan model

Machine learning lifecyle adalah sebuah *workflow* atau alur kerja pengembangan model ML dimulai dari proses pengumpulan data sampai model tersebut siap digunakan. *Machine learning lifecycle* bersifat *forward* (maju), dan dapat berulang (iteratif) karena setiap perulangan yang dilakukan bertujuan untuk terus meningkatkan keakuratan dan performa model.

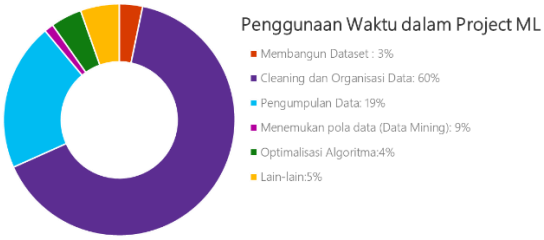
Secara umum, terdapat empat kegiatan utama dalam *Ml Lifecyle* diantaranya adalah pengumpulan data (*Data Acquisition*); Persiapan data (*Data Preprocessing);* Pelatihan , Ujicoba, Validasi dan Optimalisasi

Model(*Training, Testing, Validation* dan *Validation*); serta *Deployment Model.* Kegiatan-kegiatan tersebut dapat disusun menjadi sebuah siklus seperti pada Gambar 3.1.



Gambar 3.1. Machine Learning LifeCyle

Menurut survei yang dilakuan forbes pada tahun 2016 menunjukkan bahwa pengembang *data scientists* dan *machine learning* menghabiskan waktu sekitar 19% untuk mengumpulkan data dan 60% untuk membersihkan dan mengorganisasi data yang telah dikumpulkan. Artinya kegiatan yang paling berat dalam *Machine Learning LifeCyle* adalah pda langkah pertama dan kedua. Setelah mendapatkan data yang bersih 3% dari waktunya digunakan untuk membangun training set dari data dan 9% untuk melakukan training. Jadi dapat disimpulkan 80% waktu digunakan untuk persiapan dan 10% untuk melakukan training dan evaluasi.



Gambar 3.2. Apa yang menyita waktu data saintis?
(sumber: Forbes.com)

Adapun penjelasan langkah-langkah dalam ML Life cycle adalah sebagai berikut:

3.1 Pengumpulan Data (*Data Aquisition*)

Pengumpulan data atau (*Data Acquisition)* merupakan tahap paling awal pada proses ini. Pengumpulan dapat dilakukan pada sistem yang sedang berjalan, mulai dari data transaksi, ataupun perilaku pengguna seperti data transaksi penjualan, data pencarian barang, data aktifitas pencarian data dan lain-lain.

Langkah pertama yang harus dilakukan adalah menentukan spesifikasi kebutuhan data. Spesifikasi kebutuhan data berisi poin-poin yang harus tersedia pada data yang akan kita kumpulkan. Dalam menentukan Spesifikasi, kita dapat memulai dengan menjawab pertanyaan-pertanyaan berikut ini

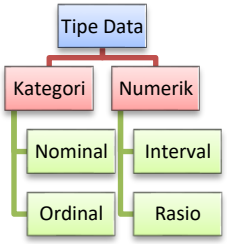
- Data apa yang akan dibutuhkan?
- Apa format data yang tersedia dan kita butuhkan?
- Dimana kita akan mendapatkan data tersebut?
- Apakah data telah tersedia secara gratis?
- Bagaimana cara mengumpulkan data tersebut?
- Bagaimana data tersebut akan digunakan?
- Seberapa besar data yang dibutuhkan?
- Apakah ada standar khusus yang harus dipenuhi oleh data tersebut?
- Bagaimana data itu disimpan dan diolah?

Data biasanya memiliki tipe dan sebelum melakukan analisis data, kita perlu mengetahui tipe dari data yang akan kita analisis. Hal ini mempengaruhi metode apa yang sesuai untuk data tersebut. Sebagai contoh, perhatikan tabel 3.1. yang merupakan sebuah dataset (kumpulan data) yang menunjukkan kondisi atau keputusan seseorang yang akan bermain tenis. Keputusan orang tersebut bermain tenis tergantung pada empat variabel *outlook, temperature, humidity, windy*. Keempat variabel ini disebut fitur

(*feature*). Setiap fitur memiliki atribut nilai dengan tipe data dan range tertentu.

Menurut pendekatan statistik, terdapat beberapa tipe atribut data diantaranya:

1. **Data Nominal**. Data ini termasuk data kategorikal, yang biasanya digunakan untuk menggambarkan/merepresentasikan karakteristik. Ia dapat digunakan untuk merepresentasikan jenis kelamin, cuaca, bahasa dan lain-lain. Nilai atribut bertipe nominal tersusun atas simbol-simbol yang berbeda, yaitu suatu himpunan terbatas. Sebagai contoh, fitur outlook pada table 3.1 memiliki tipe data nominal yaitu nilainya tersusun oleh himpunan yaitu { sunny, overcast, rainy}. Pada tipe nominal, tidak ada urutan ataupun jarak antar atribut. Tipe ini sering juga disebut kategorial atau enumerasi. Secara umum, tipe output pada supervised learning adalah data nominal.
2. **Data Ordinal**. Data ini mirip dengan data nominal, perbedaannya adalah data ini memperhatikan memiliki urutan. Urutan dari nilai memiliki peran yang penting dan signifikan. Sebagai contoh pada penilaian kepuasan. Pilihan ke-1: Sangat Puas, ke-2 : Puas, ke-3:Kurang Puas, ke-4: Tidak Puas. Kita mengetahui bahwa pilihan ke-1 lebih baik dari pilihan ke-2 atau pilihan ke-4 tetapi kita tidak dapat menilai seberapa baiknya. Contoh lainnya adalah data tingkat pendidikan dimana nilainya tersusun oleh himpunan 1-Sekolah Dasar, 2-Sekolah Menengah, 3-Diploma, 4-Sarjana, 5-Pascasarjana.



Gambar 3.3. Pembagian Tipe Data Statistik

3. **Data Interval.** Tipe interval memiliki urutan dan range nilai yang sama. Sebagai contoh 1-5, 4-10, dst. Kita dapat mentransformasikan/ mengkonversi nilai numerik menjadi nominal dengan cara merubahnya menjadi interval terlebih dahulu. Lalu, kita dapat memberikan nama (simbol) untuk masing-masing interval. Misalkan nilai numerik dengan range 1−100 dibagi menjadi 5 kategori dengan masing-masing interval adalah {1−20, 21−40, . . . , 81−100}. Setiap interval kita beri nama, misal interval 81 − 100 diberi nama nilai A, interval 61 − 80 diberi nama nilai B.
4. **Data Ratio.** Tipe *ratio* (rasio) didefinisikan sebagai perbandingan antara suatu nilai dengan nilai lainnya, misalkan massa jenis (fisika). Pada tipe ratio terdapat *absolute zero* (semacam *ground truth*) yang menjadi acuan, dan *absolute zero* ini memiliki makna tertentu**.**

Tabel 3.1. Contoh dataset play tenis (sumber:UCI machine learning)

Id	Outlook	Temperature	Humidity	Windy	Play (Kelas/Label)
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

Pada ML, data yang dibutuhkan tergantung dari jenis *task* yang akan dilakukan. Pada kasus *supervised learning*, kita akan membutuhkan data yang memiliki label. Label adalah salah satu fitur dari data yang menjadi target

kelompok atau nilai. Proses pemberian label pada data disebut data labeling. Sebagai contoh, keputusan untuk bermain (pada kolom *play*) disebut sebagai *label* atau kelas (*class*). Label tersebut menetukan apakah seorang anak akan bermain atau tidak, berdasarkan data pada kolom *Outlook, temperature, Humidity* dan *Windy*. Pada *unsupervised learning*, informasi kolom play tidak diketahui, kita harus mengelompokkan data tersebut sesuai dengan fitur-fitur yang ada.

Berdasarkan sumber datanya, data dapat dibagi menjadi 3 kelompok yaitu

- **Data Transaksi.** Data ini bersumber dari kegiatan transaksi setiap hari yang bersumber dari sistem-sistem yang telah dibangun seperti ERP (*Enterprise Resource Planning*), SCM (*Suppy Chain Management*), CRM (*Customer Relationship Management)* dan sistem informasi lainnya. Data yang dihasilkan merupakan catatan transaksi antara sistem ke sistem atau pengguna ke sistem. Contohnya data penjualan barang, data pembayaran bank, data peminjaman buku, data pembeli dan lain-lain. Data transaksi cukup sederhana dan cenderung terstruktur dengan baik. Biasanya data-data ini disimpan dengan baik dalam basis data tertentu dan dimanfaatkan sehari-hari.
- **Data Interaksi.** Data interaksi merupakan data penggunaan sistem pada suatu runtutan waktu. Misalnya data bagaimana seseorang memilih sebuah produk mulai dari pencarian sampai pembayaran. Data analisis ini biasanya digunakan untuk menganalisis kebiasaan pengguna terhadap sistem. Contohnya data perilaku pengguna, postingan sosial media, A/B test dan lain-lain.
- **Data Observasi.** Berkembangnya teknologi membuat data observasi ini dimungkinkan saat ini. Sederhananya, data observasi adalah data yang dihasilkan oleh mesin melalui sensor. Data-data tersebut dapat berupa data sensor angin, penghitung jumlah mobil, *click stream*, atau data rekaman audio dan video yang pada umumnya merupakan konten yang tidak terstruktur.

Setelah dikumpulkan, data-data tersebut harus disimpan. Mekanisme penyimpanan data ini juga harus menjadi perhatian karena bisa jadi kita akan menghadapi data yang jumlahnya sangat besar sehingga butuh mekanisme penyimpanan yang khusus seperti database, data warehouse dan sebagainya. Pemilihan mekanisme penyimpanan juga berpengaruh kepada performa mesin yang akan dibangun. Sebagai ilustrasi, kita tidak mungkin menyimpan data yang berukuran Gigabyte dalam sebuah file CSV karena proses *retrival*-nya akan memakan waktu yang lama.

Selain mengumpulkan data sendiri, kita juga dapat menggunakan dataset-dataset yang telah dikumpulkan pihak lain. Data dari pihak lain dapat kita perolehi dari beberapa sumber, seperti pihak yang menyediakan data secara terbuka sehingga diakses melalui sebuah *application programming interface* (API) seperti Twitter, Facebook, Instagram dan lain-lain dan data peneliti yang telah mempublikasikan dataset yang telah mereka kumpulkan kepada publik. Data tersebut dapat digunakan untuk belajar ML atau sebagai *data training* untuk model yang akan kita bangun. Adapun tempat favorit penulis untuk menemukan *dataset* adalah

1. Kaggle Datasets

   Kaggle merupakan salah satu tempat data set yang bagus. Dataset-dataset tersebut merupakan kontribusi dari komunitas berdasarkan data asli. Selain menyediakan tempat untuk mengunduh *dataset*, kita juga dapat berdiskusi dan melihat contoh-contoh solusi pemecahan kasus sehingga sangat direkomendasikan bagi yang sedang belajar ML.

2. Amazon Datasets

   Dataset Amazon ini menyediakan data dari berbagai topik yang berbeda misalnya transportasi publik, data ekologi, data gen, citra satelit, data perdagangan dan lain-lain. Dataset tersebut disimpan dalam Amazon Web Services (AWS)

3. UCI Machine Learning Repository

Repository dataset ini dikelola oleh University of California, School of Information and Computer Science dengan jumlah dataset lebih dari 100 dataset. Dataset-dataset tersebut telah diklasifikasikan berdasarkan tipe-tipe permasalahan masing learning misalnya kasus klasifikasi dan regresi. Selain itu kita juga dapat menemukan dataset untuk univariate dan *multivariate time-series*, atau *recommendation systems*.

4. Google's Datasets Search Engine

   Pada akhir 2018, google meluncurkan layanan terbaru untuk yang memungkin pengguna mencari dataset. Tujuannya adalah menyatukan repository-repository yang ada dan membuatnya mudah dicari.

5. *Awesome Public Datasets*

   Ini adalah kumpulan dataset yang telah dikategorikan berdasarkan topik-topik seperti Biology, Ekonomi, Edukasi dan lain-lain.

6. *Computer Vision Datasets*

   Jika anda sedang membuat penelitian dibidang *computer vision*, *deep learning* maka dataset ini cocok untuk anda. Dataset ini berisi Visual Data yang sangat berguna karena berisi objek-objek seperti *Semantic Segmentation, Image captioning, Image Generation* bahkan *Self-driving cars dataset*.

Jika data yang kita butuhkan berupa data yang tersedia di web maka, kita dapat membangun sebuah script untuk melakukan scrapping. Scrapping adalah sebuah teknik dengan membuat skript yang akan menjelajahi halaman web tertentu lalu mengekstrak informasi yang kita butuhkan. Biasanya cara in dilakukan karena *website* tertentu tidak memberikan akses API ataupun memiliki API yang terbatas. Studi kasus 1 dan 2 akan membahas bagaimana melakukan pengumpulan data dengan API dan Scrapping

3.2 Persiapan Data

Biasanya, 80% dari waktu pengembang habis untuk kegiatan ini dan hal tersebut berbanding lurus dengan hasil yang akan didapat. Seperti kata pepatah:

> *Garbage in, garbage out. (Sampah yang masuk, Sampah yang keluar)*

Semua algoritma ML dapat menemukan pola dalam data, namun apabila algoritma tersebut diberikan data yang "kacau" maka model yang dihasilkan akan memberikan luaran yang kacau juga. Jadi syarat utamanya adalah data yang bersih, bukan tentang algoritma yang dipakai. Algoritma yang sederhana tetapi dengan data yang bersih dapat mengalahkan algoritma yang rumit dengan data yang kacau.

Pemrosesan data terutama proses pembersihan data, pengembang harus menjadi detektif yang menelusuri data untuk menemukan kesalahan seperti:

- Apakah ada data yang kosong? Kolom yang tidak berisi data? Kenapa data tersebut hilang?
- Bagaimana distribusi data? Bagaimana korelasi antara kolom?
- Menyelidiki nilai-nilai yang aneh dalam data seperti "Tanggal Lahir: Pekanbaru", "No Telepon: ABCD" dan lain-lain
- Apakah isi data konsisten? Misalnya jenis kelamin bernilai "Laki-Laki", "Pria","Man","LK" dan sebagainya

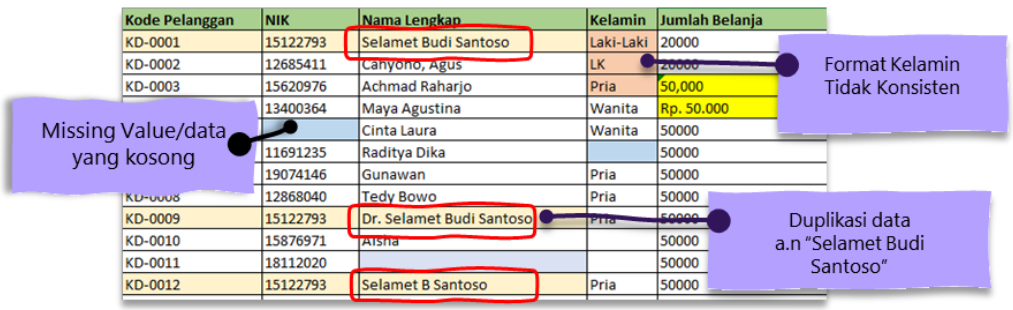
Secara umum proses persiapan terhadap data terbagi atas tiga kegiatan utama yaitu **pembersihan data (*data cleaning*), rekayasa fitur (*feature enginering*)** dan **pembagian data (*data Split*).** Dasar pengelompokan kegiatan tersebut adalah bentuk proses yang dilakukan pada data. Pada pembersihan data, operasi yang dilakukan merupakan operasi perbaikan pada level baris. Sedangkan rekayasa fitur merupakan kegiatan perbaikan data pada level kolom. Umumnya, kegiatan pembersihan data dilakukan terlebih dahulu lalu merekayasa fitur jika diperlukan. Namun pada kasus-

kasus tertentu dapat dilakukan rekayasa fitur terlebih dahulu. Jadi urutan pelaksanaan disesuaikan dengan kasus yang dihadapi.

Kegiatan-kegiatan ini dilakukan agar kita memiliki data yang baik yang benar-benar merepresentasikan kasus atau permasalahan yang sedang dihadapi. Penggunaan data mentah dan membangun model di atas data ini secara langsung akan mengakibatkan model yang dihasilkan memiliki performa yang tidak maksimal atau tidak dapat bekerja seperti diinginkan. Hal ini disebabkan algoritma yang digunakan tidak memiliki kemampuan untuk mengekstraksi fitur secara otomatis. Peran manusia dibutuhkan untuk mengolah data mentah tersebut sehingga dapat dihasilkan data yang lebih "siap" untuk di proses.

3.2.1 Pembersihan Data

Pembersihan data atau *Data Cleaning* adalah proses mendeteksi dan memperbaiki (atau menghapus) data yang rusak atau tidak akurat dari dataset, tabel, atau basis data untuk meningkatkan kualitas data. Pada tahap ini juga dilakukan identifikasi terhadap data yang tidak lengkap, salah, tidak akurat atau tidak relevan dari dataset. Perbaikan terhadap data dapat dilakukan dengan mengganti, memodifikasi, atau menghapus data kotor tersebut. Pembersihan data dapat dilakukan secara interaktif dengan tool *data wrangling*, atau melalui pemrosesan batch melalui skrip. Gambar 3.4 menunjukkan beberapa permasalahan pada data (terduplikasi, kosong dan tidak memiliki format yang sama).



Gambar 3.4. Contoh Dataset Kotor, berisi data yang terduplikasi, kosong, inkonsistensi, berbeda format penulisan dan lain-lain

Setelah pembersihan, dataset diharapkan menjadi konsisten. Inkonsistensi yang terdeteksi atau dihapus dapat disebabkan oleh kesalahan entri pengguna, oleh kesalahan dalam pengiriman atau penyimpanan data, atau oleh kamus data yang berbeda dari entitas yang sama di sumber yang berbeda. Tujuan dari pembersihan data adalah meningkatkan kualitas data.

Kualitas data ditunjukkan oleh beberapa indikator berikut ini:

1. Validitas

Kita sering mendengar pertanyaan "Datanya valid nggak?". Pertanyaan itu sebenarnya ingin menanyakan tentang kebenaran data yang dimiliki. Kebenaran sebuah data diukur dengan sejauh mana data sesuai dengan aturan atau batasan bisnis yang ditetapkan. Sebagai contoh data tanggal lahir harus diisi dengan sebuah data tanggal yang lebih kecil atau sama dengan tanggal hari ini.

Dalam komputasi, ada istilah *Data constraints* (batasan data). Batasan data inilah yang dapat digunakan untuk menjamin validitas data. *Data constraints* dibagi menjadi beberapa jenis diantaranya:

1. *Data-Type Constraints* – Batasan pada tipe data pada kolom penyimpanan data. Misalnya Total Penjualan disimpan dalam bentuk *floating point,* nama disimpan dalam tipe string, tanggal disimpan pada tipe data *date* dan lain-lain. Tipe data yang digunakan untuk menyimpan data harus sesuai dengan isi dari data tersebut.
2. *Range Constraints*. Biasanya batasan ini berlaku untuk angka atau tanggal berada dalam batasan tertentu seperti nilai minimum dan atau maksimum sebuah nilai. Contoh nyatanya adalah nilai pembayaran tidak boleh <0.
3. *Mandatory Constraints*. Batasan ini memiliki arti kolom tertentu tidak boleh kosong. Misalnya kolom nama memiliki *constrain mandatory,* ini berarti kolom nama tidak boleh dikosongkan.

4. *Unique Constraints*: Sebuah atau beberapa *field* harus memiliki sebuah kolom yang bersifat unik. Contohnya 2 orang tidak boleh memiliki NIK (Nomer Induk Kependudukan) yang sama.
5. *Set-Membership constraints*: Batasan berupa nilai-nilai sebuah kolom berasal dari sekumpulan data atau kode yang statis. Statis artinya kecil kemungkinan untuk menambah anggota *membership*-nya. Contohnya jenis kelamin harus diisi Laki-Laki atau Perempuan.
6. *Foreign-key constraints*: Batasan ini mirip dengan *Set-Membersip*, tetapi nilai data disimpan didalam tabel lain dan dapat bersifat umum serta dinamis. Sebagai contoh kolom kode kecamatan, nilainya terbatas pada isi tabel kecamatan.
7. *Regular expression patterns*: Batasan ini biasanya ada pada tipe text, dimana sebuah teks harus mengikuti pola-pola penulisan tertentu. Contohnya adalah Email. Dimana sistem akan menolak setipa nilai yang tidak sesuai dengan format yang ditentukan

2. Akurasi

Akurasi menunjukkan tingkat kesesuaian dengan sebuah nilai standar atau nilai sebenarnya. Akurasi **sangat sulit dicapai** melalui proses *data-cleansing* secara umum karena biasanya nilai standar atau kebenaran sebuah data tidak ada seperti data tanggal lahir, nama dan lain-lain. Akurasi dapat dicapai pada proses *cleansing* ketika kita mengetahui sebuah nilai kebenaran diketahui contohnya nama kota, ketika ada kota "Jakarte", maka kita mengetahui bahwa data itu salah/tidak akurat karena kita memiliki referensi data sebenarnya.

3. Kelengkapan

Kelengkapan artinya tingkat dimana semua aspek data diketahui. Ketidaklengkapan hampir tidak mungkin untuk diperbaiki dengan metodologi pembersihan data karena seseorang tidak dapat menyimpulkan fakta yang tidak ditangkap/direkam ketika proses pengambilan data.

4. Konsistensi

Konsistensi merupakan tingkat dimana semua sebuah aspek pengukuran (bisa kolom tertentu) memiliki nilai yang sama atau setara didalam semua sistem. Sebagai contoh data email pelanggan yang direkam diaplikasi A harus sama dengan email yang berada di Aplikasi B. Memperbaiki data yang tidak konsisten bukanlah hal yang mustahil. Beberapa strategi dapat dilakukan misalnya mengambil daya yang terbaru sebagai data yang akan digunakan atau mencari kebenaran tentang data tersebut (dengan menelepon pelanggan).

5. Keseragaman

Data yang digunakan memiliki unit pengukuran yang sama dalam semua sistem. Misalnya data berat menggunakan Kilogram, artinya semua data yang menggunakan satuan berat lainnya misalnya gram dan pound harus dikonversi ke Kilogram.

Adapun tahap-tahap yang harus dilakukan dalam proses pembersihan data dengan tujuan untuk menghasilkan data dengan kualitas yang baik. Adapun masalah-masalah umum yang ada pada data adalah *Missing Values, Duplicate data, Invalid data*, dan *Noise*.

Adapun langkah-langkah pembersihan data adalah

Langkah 1. Inspeksi terhadap data

Proses meliputi pendeteksian data yang tidak diharapkan, tidak konsisten, tidak valid, korelasi data serta mempelajari distribusi data. Proses ini adalah proses yang melelahkan, memakan waktu yang lama dan membutuhkan banyak metode dalam pendeteksian error pada data. Berikut ini adalah teknik-teknik yang dapat digunakan untuk menemukan error pada data.

- **Data profiling**

  Proses menghasilkan sebuah ringkasan statistik tentang data dapat disebut data *profiling*. Ringkasan tersebut sangat membantu kita

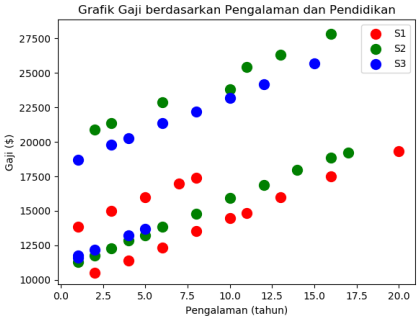
untuk mendapatkan informasi tentang kualitas dari data. Informasi yang ditemukan tersebut mencakup aspek struktur, isi dan relasi. Aspek struktur membantu untuk menentukan apakah data kita konsisten dan terformat dengan baik. Aspek isi fokus kepada kualitas data terkait format, standar dan integritas data, Sebagai contoh alamat email harus sesuai dengan format email secara umum (nama@alamat.com).

Data profiling dilakukan dengan menggunakan beberapa teknik statistika desktriptif termasuk mean (rata-rata), minimum, maximum, percentile, frekuensi dan fungsi agregat lainnya seperti count dan sum. Tambahan informasi mengenai data tentang data (metadata) juga menjadi keluaran yang sangat penting seperti tipe data, ukuran kolom (*length*), discrete values, nilai unik dan lain-lain. Biasanya pada tahap ini dilakukan analisa kelengkapan data (*Completeness Analysis)* untuk mengetahui seberapa banyak atribut yang diberikan digunakan, kosong atau nol; Analisa keunikan data (*Uniqueness Analysis*) untuk mengerahui berapa banyak kolom nilai unik (distinct) yang ditemukan untuk semua atribut? Apakah ada duplikat? Haruskah ada attribut unik tersebut ada; Analisa distribusi nilai (*Values Distribution Analysis*) untuk mengetahui distribusi data yang akan digunakan; *Range Analysis* bertujuan untuk menemukan nilai minimum, maximum, average dan median dari attribut data; Sehingga dengan analisa-analisa tersebut, dapat ditemukan beberapa informasi penting, misalnya apa tipe data yang digunakan? Berapa banyak data yang hilang? Bagaimana distribusi data? Apakah ada relasi antar data?

- **Visualisasi Data**

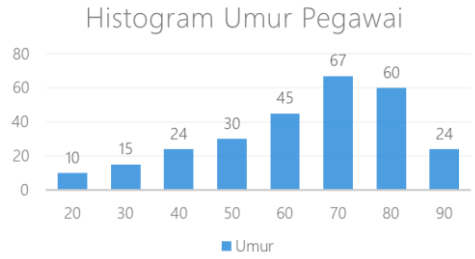
  Dengan melakukan analisa dan visualisasi data, informasi tentang data dapat dilihat secara visual. Visualisasi dapat dilakukan menggunakan grafis seperti scatter plot, histogram, dotplot, boxplot dan lain-lain.

Scatter plot umunya digunakan untuk memvisualiasaikan relasi antara dua variabel atau attribut numerik. Sebagai contoh perhatikan Gambar 3.5, kita dapat mengidentifikasi bahwa variabel pengalaman kerja diduga mempengaruhi variabel lain yaitu gaji. Dengan meletakkan variabel pengalaman sebagai *explanatory variable* pada sumbu x dan variabel gaji sebagai *response variable* pada sumbu Y, kita dapat melihat bagaimana relasi kedua variabel tersebut, apa saja pencilan/outlier yang ada. Dengan memvisualiasikannya maka kita mendapatkan informasi baru yang berharga sehingga dapat digunakan untuk memperbaiki atau meningkatkan akurasi model.



Gambar 3.5. Visualisasi dalam bentuk Scatter Plot

Contoh lain visualisasi yang sering digunakan adalah histogram. Histrogram adalah salah satu cara yang bagus untuk memvisualisasikan distribusi variabel numerik. Data dikelompokkan pada interval tertentu dan tinggi bar merepresentasikan jumlah item per interval. Selain itu, histogram juga dapat digunakan untuk meidentifikasi bentuk distribusi. Gambar 3.6 adalah contoh histogram. Distribusi data *life expectancies* kelihatan condong kesebelah kiri (*left skewed*) diantara umur 60 sampai 90 tahun.

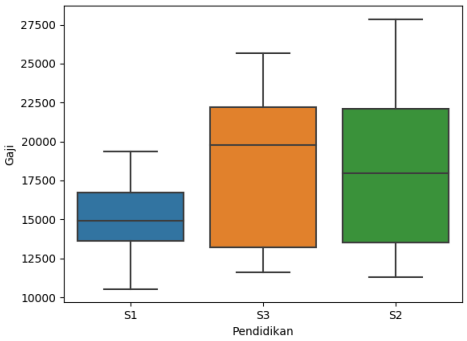


Gambar 3.6. Visualisasi dalam bentuk Histogram

Selain itu, kita dapat menggunakan *Boxplot*. *Boxplot* merupakan ringkasan distribusi sampel yang disajikan secara grafis yang bisa menggambarkan bentuk distribusi data (*skewness*), ukuran tendensi sentral dan ukuran penyebaran (*keragaman*) data pengamatan. Terdapat 5 ukuran statistik yang bisa kita baca dari boxplot, yaitu:

- nilai minimum: nilai observasi terkecil
- Q1: kuartil terendah atau kuartil pertama
- Q2: median atau nilai pertengahan
- Q3: kuartil tertinggi atau kuartil ketiga
- nilai maksimum: nilai observasi terbesar.

Selain itu, *boxplot* juga dapat menunjukkan ada tidaknya nilai *outlier*/pencilan dan nilai ekstrim dari data pengamatan. Bagian utama *boxplot* adalah kotak berbentuk persegi (*Box*) yang merupakan bidang yang menyajikan *Interquartile range* (IQR), dimana 50 % dari nilai data pengamatan terletak di sana. Panjang kotak sesuai dengan jangkauan kuartil dalam (*inner Quartile Range*, IQR) yang merupakan selisih antara Kuartil ketiga (Q3) dengan Kuartil pertama (Q1). IQR menggambarkan ukuran penyebaran data. Semakin panjang bidang IQR menunjukkan data semakin menyebar. Gambar 3.7 menunjukkan contoh grafik boxplot yang menggambarkan kondisi gaji berdasarkan jenjang pendidikan.



Gambar 3.7. Visualisasi Menggunakan Boxplot

Berdasarkan Gambar 3.7, rata-rata Pendidikan S1 memiliki gaji 15.000, S2 memiliki rata-rata gaji 175.00 dan S3 memiliki rata-rata gaji 20.000.

Langkah 2. Pembersihan data / *Data Cleaning*

Memperbaiki *anomaly* yang ditemukan pada proses inspeksi. Pembersihan data dilakukan dengan berbagai cara berdasarkan permasalahan yang dihadapi dan tipe data. Secara umum aktifitas yang dilakukan pada pembersihan data adala penghapusan, perbaikan, atau penghitungan ulang. Berikut ini adalah kondisi-kondisi dimana data perlu dibersihkan

1. **Data yang tidak relevan**

   Data yang tidak relevan dapat dikatakan data yang tidak dibutuhkan atau tidak cocok untuk konteks permasalahan yang dihadapi. Sebagai contoh, ketika kita ingin membuat model yang dapat menentukan apakah seorang pasien terkena kangker atau tidak maka attribut nomer telepon pasien tidak dibutuhkan dan tidak relevan dengan kasus yang akan diselesaikan. Pada kasus ini, kita harus membuang kolom/attribut/fitur nomer telepon karena tidak dibutuhkan.

   Proses pemilihan kolom tersebut *feature selection* atau seleksi fitur. Proses seleksi fitur ini dapat dilakukan secara manual (dengan mengetahui konteks permasalahan) dan secara otomatis menggunakan algortima tertentu. Beberapa algortima yang dapat digunakan adalah *Univariate Selection, Recursive Feature Elimination, Principal Component Analysis,*

Feature Importance, dan lain-lain. Beberapa algoritma berbasis pohon keputusan dan neural network telah melakukan proses ini secara otomatis sebagai bagian dari algoritma seperti pada Algoritma Random Forest.

Pembersihan data yang tidak relevan tidak hanya dilakukan pada level kolom atau attribut, tetapi juga dapat dilakukan pada level baris. Sebagai contoh, anda ingin membangun sebuah model dapat memprediksi kebakaran hutan didaerah Provinsi Riau maka anda harus menghapus data yang berada di luar area Provinsi Riau karena data tersebut juga tidak relevan dengan model yang akan dibangun.

Keuntungan melakukan seleksi fitur adalah mengurangi *Overfitting* dan meningkatkan akurasi karena dengan berkurangnya data yang tidak relevan berarti mengurangi kemungkinan untuk membuat model berdasarkan data kotor/*noise*, sehingga model yang dihasilkan lebih akurat. Selain itu, seleksi fitur juga mengurangi waktu pelatihan. Lebih sedikit data yang diproses akan mengurangi kompleksitas dan waktu pelatihan akan lebih cepat.

2. Data yang terduplikasi

Duplikasi data banyak terjadi didunia nyata. Data yang terduplikasi adalah ada yang sama atau berulang pada dataset. Penyebab duplikasi ada banyak misalnya karena data dikumpulkan dari berbagai sumber data atau pada saat pengisian menekan tombol simpan berkali-kali. Untuk mengatasi data yang terduplikasi harus di tentukan identitas data yang terduplikasi dan menghapus data yang terduplikasi tersebut.

3. Konversi Tipe Data

Pastikan data angka disimpan dalam tipe data numerik, data tanggal disimpan dalam tipe data *date* atau *timestamp* dan seterusnya. Tipe data ini telah terdeteksi pada tahapan pertama (ketika inspeksi). Jika dibutuhkan data yang bersifat kategori dapat dikonversi menjadi bentuk angka.

4. Perbaikan dan Modifikasi Data

Banyak kemungkinan kesalahan yang terjadi pada data misalnya salah ketik, data yang tidak konsisten dan lain-lain. Berikut ini beberapa kesalahan yang umum terjadi pada data:

Tambahan *White Space*. Istilah *white space* adalah karakter-karakter yang tidak kelihatan pada data seperti spasi dan tab. Data dengan nilai " Halo " (ada spasi di depan dan di belakang) tidak sama dengan data "Halo" tanpa spasi. Jadi jika anda memiliki misalnya data yang sensitif maka perlu melakukan penghilangan *whitespace*. Berikut adalah contoh penghilangan *whitespace* pada Python

Source Code - Implementasi Pembersihan White Space

```
1. data = ' Saya memiliki white space    '
2. print(data)
3. print('Data sesudah')
4. print(data.strip())
```

Keluaran

```
 Saya memiliki white space
Data sesudah
Saya memiliki white space
```

***Pad strings*:** Untuk tujuan mempermudah pembacaan biasanya angka-angka tertentu diberikan awalan 0 (nol) seperti data bulan Januari biasanya ditulis 01. Jika kita membutuhkan data dalam bentuk Integer maka perlu menghilangkan angka nol di depan dan mengkonversi tipe data.

Source Code 1. Implementasi Pembersihan Padded String

```
1. data = '00001'
2. print(data)
3. print('Data sesudah')
4. print(data.strip())
```

Keluaran

```
00001
Data sesudah
1
```

Salah ketik. Salah ketik adalah kesalahan yang paling umum karena data string dapat diisi dalam berbagai jenis. Sebagai contoh jenis kelamin

laki-laki dapat ditulis dengan banyak kemungkinan LK, Laki-laki, Laki-Laki, LAKI, LAKI-LAKI, Lelaki, leki-leki dan lain-lain. Untuk menghadapi masalah ini langkah pertama adalah membuat sebuah barplot yang berisi semua nilai unik dari kelamin laki-laki. Selanjutnya baru melakukan *replace* terhadap pola-pola tesebut.

5. **Standarisasi data**

   Selain pembersihan, standarisasi data perlu dilakukan untuk menjamin bahwa data yang akan digunakan terdiri atas format yang standar. Misalnya untuk data *string* harus dipastikan semua nilainya berjenis hurup kecil atau huruf besar semua. Sedangkan untuk data numerik, pastikan data-data tersebut memiliki satuan ukur yang sama misalnya untuk Panjang semua data disimpan dalam centimeter semua atau inci semua. Selain itu pastikan juga untuk data tanggal memiliki format yang sama missal YYYY-MM-DD atau DD/MM/YYYY.

6. **Perbaikan Missing data**

   Faktanya, data yang didapat sering mengalami *missing value* atau data yang tidak lengkap. *Missing data* atau data yang kosong pada sebuah dataset yang berasal dari kejadian yang bersifat acak. Missing data perlu ditangani dengan baik karena beberapa algoritma tidak bisa bekerja jika pada dataset berisi *missing data*. Berikut ini beberapa pilihan yang dapat dilakukan jika pada dataset memiliki *missing data* diantaranya:

   - Hapus data yang memiliki *missing data*. Penghapusan ini dilakukan jika data tersebut jumlahnya tidak banyak. Sebagian literatur mengatakan bahwa jika jumlah data yang hilang kurang dari 2% maka opsi ini dapat dilakukan.
   - Mengganti nilai yang hilang dengan nilai rata-rata atau median atau modus.
   - Menggunakan metode-metode regresi atau ML untuk memprediksi nilai yang hilang.

7. Transformasi data

Transformasi Data adalah upaya yang dilakukan dengan tujuan utama untuk mengubah skala pengukuran data asli menjadi bentuk lain sehingga data dapat memenuhi asumsi-asumsi yang mendasari analisis ragam. Transformasi data ada beberapa jenis, antara lain:

- Transformasi *Square Root* (Akar),
- Transformasi Logaritma,
- Transformasi *Arcsin*,
- Transformasi *Square* (Kuadrat),
- Transformasi *Cubic* (Pangkat Tiga),
- Transformasi Inverse (Kebalikan),
- Transformasi *Inverse Square Root* (Kebalikan Akar),
- Transformasi *Inverse Square* (Kebalikan Kuadrat),
- Transformasi *Inverse Cubic* (Kebalikan Pangkat Tiga),
- Transformasi *Reverse Score* (Balik Skor).

8. Normalisasi data

Tujuan utama dari normalisasi adalah mengubah nilai numerik dari sebuah fitur pada *dataset* menjadi sebuah nilai yang lebih umum. Perubahan tersebut biasanya dilakukan pada kolom yang memiliki jangkauan nilai yang berbeda, namun perubahan nilai tersebut tidak mengubah makna dari data. Tujuannya adalah agar model yang dihasilkan lebih stabil. Manfaat yang didapat adalah

Data menjadi *Comparable*. Misalnya seorang pembeli melakukan pembelian sebesar 3.000.000 untuk 6 item barang. Dengan menormalisasikan nilainya, maka kita dapat membandingkan kedua kolom tersebut. Nilai pembelian menjadi 0.8 dan jumlah item menjadi 0.2. Masing-masing kolom

tersebut memiliki nilai minimal 0 dan maksimal 1, jadi transaksi tadi merupakan transaksi besar, namun jumlah pembelian sedikit.

Selanjutnya, data yang ternormalisasi mempercepat algoritma untuk konvergen sehingga waktu pelatihan dapat dikurangi. Dan bagi algoritma-algoritma tertentu uang membutuhkan distribusi normal maka hal ini dapat membantu algoritma bekerja lebih baik. Beberapa teknik yang umum digunakan adalah *scaling, clipping* dan *Z-Score*.

3.2.2 Rekayasa Fitur

Rekayasa fitur atau *Feature Enginering* adalah proses perubahan dataset dimana dilakukan proses ekstraksi sebuah kolom menjadi kolom-kolom lainnya atau proses mereduksi kolom.

Fitur pada ML diasosiasikan dengan kolom pada dataset. Sebuah kolom dapat memiliki tipe-tipe data tertentu. Sebuah fitur adalah sebuah atribut data yang berguna atau memiliki arti penting terhadap masalah yang akan diselesaikan. Sebagai contoh jumlah waktu belajar siswa merupakan atribut penting ketika kita ingin memprediksi kelulusan seorang siswa. Namun attribut nama siswa berbeda, ia tidak memberikan pengaruh kepada kelulusan siswa. Oleh karena itu pemilihan fitur yang tepat sangat mempengaruhi hasil atau performa model. Pemilihan fitur ini merupakan salah satu kegiatan yang dilakukan dalam rekayasa fitur.

Berikut ini kriteria-kriteria sebuah fitur yang baik diantaranya:

1. **Informatif**. Sebuah fitur harus memberikan informasi yang baik kepada penggunanya.
2. **Ketersediaan Data**, jika fitur ini datanya banyak yang kosong. Jika anda memilih fitur yang banyak kosong maka anda akan berurusan dengan missing data.
3. **Diskriminan**. Dengan fitur ini diharapkan dapat membagi data menjadi target kelas yang dituju atau berkorelasi dengan label.

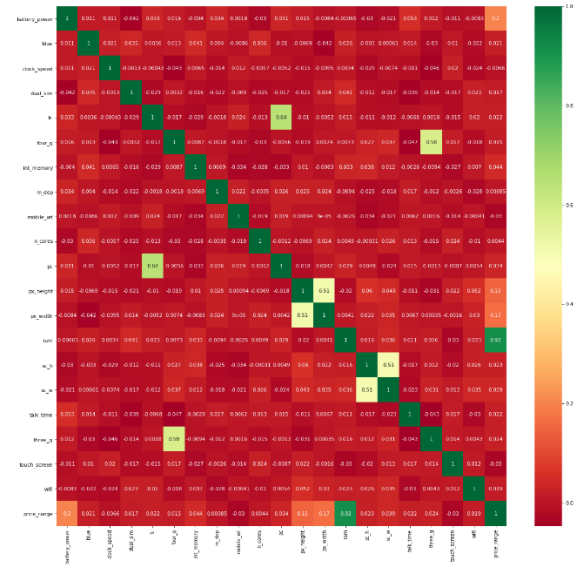
Berikut ini beberapa kegiatan yang dapat dilakukan dalam rekayasa fitur diantaranya:

3.2.2.1 FEATURE SELECTION

Pada sebuah dataset, tidak semua fitur berguna dalam pengerjaan *task* tertentu. Fitur-fitur yang tidak relevan terhadap kasus sebaiknya di buang karena kehadirannya dapat mengganggu performa model.

Berikut ini metode-metode yang sering digunakan oleh praktisi dalam melakukan seleksi fitur diantaranya :

- *Correlation Matrix*. Secara matematis, sebuah fitur dikatakan penting jika memiliki korelasi yang tinggi dengan variabel dependent. Oleh karena itu, koefisien korelasi adalah metode yang umum digunakan untuk menentukan seberapa penting sebuah fitur.



Gambar 3.8. Contoh *Correlation matrix*

- *Univariate Selection.* Kita dapat menggunakan fungsi-fungsi test statistik dalam menentukan seberapa penting kolom. Salah satu metode yang populer digunakan adalah SelectKBest menggunakan ANOVA F-value.

- Mengunakan model-model yang memiliki fitur *Feature Importance* seperti model-model yang berbasis tree seperti Random Forest. Selain digunakan untuk klasifikasi atau regresi, model model tersebut dapat menghitung tingkat *importance* (seberapa penting) sebuah fitur berdasarkan hasil pelatihan.
- *Permutation feature importance.* Pada metode ini didasarkan pada intuisi bahwa jika fitur tidak/kurang penting, maka mengubah atau mengganti atau menukar nilainya tidak akan menghasilkan penurunan yang signifikan dalam kinerja model. Namun, jika fitur penting untuk memprediksi hasil, maka perubahan tersebut akan memiliki efek yang cukup signifikan
- *Drop feature importance.* Metode ini mirip dengan *Permutation feature importance,* namun pada kolom ini bekerja dengan menghapus kolom lalu melihat efek dari penghapusan tersebut. Jika mempengaruhi hasil maka kolom dapat dinyatakan penting.
- *Feature Importance* berbasis nilai SHAP (SHapley Additive exPlanations). Nilai SHAP merupakan sebuah metode yang sangat luar biasa untuk melakukan "*reverse-enginerring*" terhadap luaran sebuah predictive model. Sebagai contoh anda memprediksi harga sebuah apartemen adalah 3 miliar dengan nilai fitur, luas apartemen 50 m^2, dekat dengan stasiun, dan tidak boleh memelihara hewan peliharaan. Anda juga di berikan informasi bahwa rata-rata harga apartemen adalah 3.5 miliar. Maka dengan menggunakan SHAP, anda dapat menentukan berapa kontribusi dari masing-masing fitur terhadap 3 milliar tersebut.

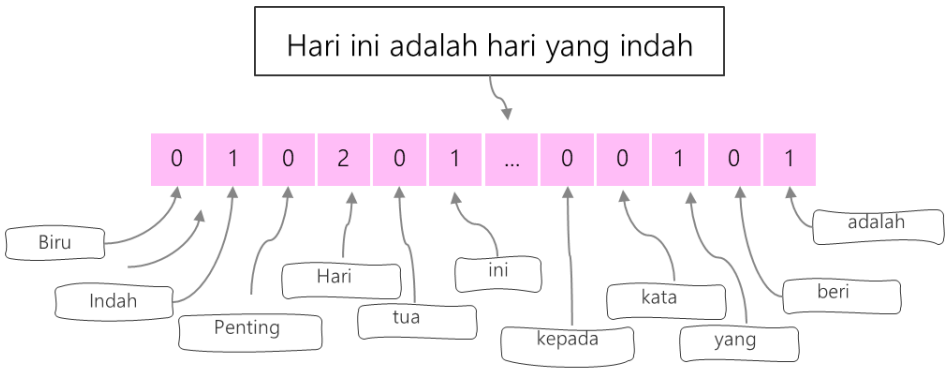
3.2.2.2 FEATURE EXTRACTION

Feature extraction adalah sebuah proses pembentukan fitur baru atau mengurangi dimensi data dari data mentah secara otomatis. *Feature extraction* biasanya dilakukan pada data tabular berdimensi tinggi, gambar, suara, dan text.

Pada data-data yang bersifat tabular (dalam bentuk tabel), biasanya data-data tersebut memiliki attribut yang sangat banyak atau dimensi tinggi. Attribut yang sangat banyak tersebut dapat dikurangi secara otomatis menggunakan algoritma tertentu seperti Principal Component Analysis, Singular Value Decomposition, Linear Discriminant Analysis, Locally Linear Embedding dan lain-lain. Kunci dari feature extraction adalah penggunaan metode yang otomatis.

Berikut ini contoh-contoh kegiatan *feature extration*:

- *Bag of Words. Bag-of-Words* (BOW) adalah teknik yang paling banyak digunakan untuk ekstraksi fitur pada kasus *natural language processing*. Sederhananya, BOW merupakan teknik yang mengubah data yang berupa teks menjadi bentuk vektor sehingga dapat dilakukan pemrosesan. Sebagai contoh sebuah kalimat "Hari ini adalah hari yang indah" akan diekstrak menjadi sebuah vektor yang berisi jumlah kata. Sebagai contoh pada gambar 3.9, pada awal vektor bernilai 0 karena pada kalimat tidak ada kata biru, sedangkan item yang kedua bernilai 1 karena ada 1 kata indah.



Gambar 3.9. Visualisasi Menggunakan Boxplot

- Ektraksi tepi pada gambar (*Image Processing*). Pemrosesan gambar merupakan salah satu domain yang banyak memanfaatkan ekstraksi fitur. Contoh operasi yang dilakukan adalah ekstraksi tepi, bentuk, atau gerak pada gambar.

- Ekstraksi Mel Spektogram pada audio (*Audio Processing*) dengan mengubah audio yang berupa frekwensi dalam waktu tertentu menjadi mel spectrogram.

3.2.2.3 FEATURE CONSTRUCTION

Feature Construction atau pembentukan fitur baru dari *raw data* (data mentah) dilakukan secara manual. Fitur hasil dari kegiatan ini dibuat langsung oleh pengembang menggunakan tool dan pengetahuan pengembang. Kegiatan ini biasanya dibantu oleh Informasi yang yang didapat dari *feature important* atau kebutuhan input model. Karena proses pembuatannya manual, kadang-kadang aktivitas ini memerlukan waktu dan pengamatan masalah secara teliti agar fitur yang dikembangkan dapat bermanfaat. Kegiatan feature construction ini sangat tergantung kepada jenis data, algoritma dan task yang akan diselesaikan.

Pada data tabular, biasanya dilakukan proses penggabungan atau pemisahan data pada kolom-kolom tertentu sehingga tercipta kolom baru. Contoh kegiatan ini adalah *Categorical Encoding* seperti *One hot encoding; Count and Frequency encoding, Target encoding / Mean encoding, Ordinal encoding, Weight of Evidence* dan lain-lain.

Selanjutnya, pada data yang berbentuk gambar kegiatan ini bervariasi tergantung permasalahan yang dihadapi. Contohnya kita akan membuat sebuah sistem yang mendeteksi lokasi sel kanker/rusak dari sekumpulan sel. Maka kegiatan untuk melakukan anotasi berupa koordinat lokasi merupakan kegiatan *Feature Construction*.

Jadi kegiatan ini bersifat manual dan butuh keahlian spesifik tetapi data yang dihasilkan akan sangat bermanfaat bagi model.

3.2.2.4 FEATURE LEARNING

Feature learning adalah proses identifikasi dan penggunaan fitur secara otomatis dari data mentah. Biasanya feature learning digunakan pada metode-metode *Representation learning* atau *Deep learning.* Contohnya adalah autoencoders dan *restricted Boltzmann machines.* Algoritma ini telah

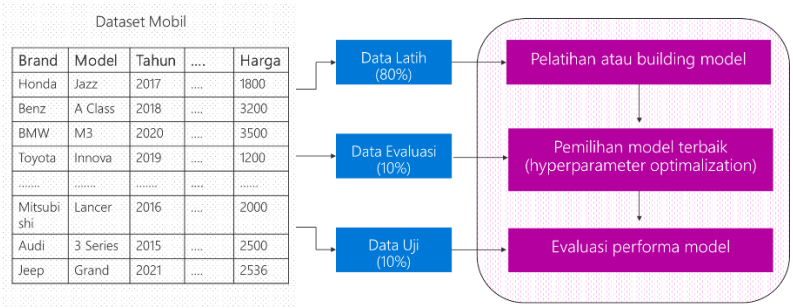
menunjukkan kemampuannya dalam menggenerate fitur secara baik dan mempelajari *abstract representations* dari fitur-fitur tersebut. Kasus-kasus seperti *speech recognition, image classification, object recognition* dan lainnya merupakan area yang tepat untuk menggunakan metode ini.

3.2.3 Data splitting

Setelah mendapatkan data yang bersih, selanjutnya adalah melakukan proses *data spliting* (pemecahan data) dimana kita membagi data tersebut menjadi beberapa bagian diantaranya data latih (*training data*), data uji (*testing data*), dan data validasi (*validation data*). Metode ini dikenal dengan nama *holdout* dimana kita membagi dataset menjadi 2 bagian yaitu *training set* dan *holdout set. Training set* berisi data latih yang digunakan untuk membangun model dan biasanya ukurannya lebih besar dibandingkan dengan *holdout set.*

Sedangkan *holdout set* dapat terdiri atas data uji (*testing data*), dan data validasi (*validation data*) atau hanya data uji saja. Dengan kata lain kita dapat memecah dataset tersebut menjadi 2 atau 3 tergantung kebutuhan. Data validasi bersifat opsional dan hanya dibutuhkan ketika kita ingin melakukan optimalisasi hyperparameter model yang telah dilatih atau pemilihan algoritma terbaik. Sebagai contoh, pada algoritma yang berbasis jaringan syaraf tiruan, dimana kita perlu menentukan hyperparameter seperti berapa jumlah *hidden unit*, jumlah layer dan sebagainya. Untuk menentukan hyperparameter tersebut maka diperlukan sebuah dataset yang tidak bias sehingga menghasilkan hyperparameter yang optimal. Namun jika pada pelatihan anda telah menetapkan hyperparameter model maka data evaluasi biasanya tidak dibutuhkan. Selanjutnya data uji digunakan untuk mengukur performa model yang telah dilatih.

Sebagai contoh pada Gambar 3.10, dimana kita memiliki sebuah dataset mobil yang terdiri atas 1.000 data jenis mobil. Untuk melakukan pelatihan model maka data tersebut akan dipecah menjadi 3 bagian. Data Latih terdiri atas 800 data, Data evaluasi 100 data dan Data Uji 100 data.



Gambar 3.10. Ilustrasi *holdout sets*

Tidak ada aturan yang pasti tentang berapa porsi pembagian masing masing jenis data. Jika kita memiliki dataset yang besar, misalnya 500.000 sample maka kita dapat menggunakan 95% untuk data latih dan 2.5% untuk validasi, dan 2.5% untuk data uji. Namun jika kita hanya memiliki 100 data kita dapat memecahnya menjadi **70+15+15**.

Kenapa dataset harus dipecah? Bayangkan jika anda melakukan ujian dengan soal yang pernah anda kerjakan maka anda akan mendapatkan nilai yang tinggi. Namun ujian tersebut tidak menggambarkan kemampuan anda yang sebenarnya. Oleh karena itu soal yang digunakan pada saat pembelajaran dan ujian harus berbeda, namun dari sumber yang sama.

Untuk melakukan split terhadap data di python menggunakan Numpy maka dapat dilihat pada contoh berikut menggunakan np.split

```
1  import numpy as np
2  from numpy import loadtxt
3  my_data = loadtxt('data/diabetes.csv', delimiter=',' ,skiprows=1)
4  train, validate, test = np.split(my_data, [int(.6*len(my_data)), int(.8*len(my_data))])
5  print("Dataset : ",len(my_data))
6  print("Train Dataset : ",len(train))
7  print("Validation Dataset : ",len(validate))
8  print("Test Dataset : ",len(test))
```

```
Dataset :  768
Train Dataset :  460
Validation Dataset :  154
Test Dataset :  154
```

Untuk melakukan split terhadap data di python menggunakan Pandas maka dapat dilihat pada contoh berikut:

```python
import numpy as np
import pandas as pd

#load dataset menggunakan pandas
df = pd.read_csv('data/diabetes.csv')

def train_validate_test_split(df, train_percent=.6, validate_percent=.2, seed=None):
    np.random.seed(seed)
    perm = np.random.permutation(df.index)
    m = len(df.index)
    train_end = int(train_percent * m)
    validate_end = int(validate_percent * m) + train_end
    train = df.iloc[perm[:train_end]]
    validate = df.iloc[perm[train_end:validate_end]]
    test = df.iloc[perm[validate_end:]]
    return train, validate, test

train, validate, test = train_validate_test_split(df)
```

Untuk melakukan split terhadap data di python menggunakan Sk-Learn maka dapat dilihat pada contoh berikut menggunakan fungsi train_test_split:

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

#load data iris
iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.15)
# 0.25 x 0.8 = 0.2
```

Manfaat lainnya yang didapatkan ketika membagi data-data ini adalah kita dapat melakukan deteksi terhadap dua masalah yang umum terjadi pada saat pembelajaran yaitu *underfitting* dan *overfitting*:

- *Underfitting* adalah sebuah situasi dimana model yang dihasilkan gagal untuk melakukan generalisasi relasi pada data. Sebagai contoh misalnya ketika kita melakukan training pada data yang non-linear menggunakan metode linear yang mengakibatkan performa dari training dan test rendah. Umumnya masalah *underfitting* diidentifikasi dengan rendahnya performa *training* dan *test sets.*

- *Overfitting* adalah sebuah situasi dimana model yang dihasilkan menghasilkan generalisasi yang kompleks dari data training. Analoginya, model yang dihasilkan "menghapal" relasi pada data sehingga ketika dievaluasi menggunakan test data maka performanya akan turun. Masalah ini diidentifikasi dengan tingginya performa pada training namun performa yang rendah pada test. Idealnya performa antara training dan test tidak jauh berbeda.

Metode *cross-validation*

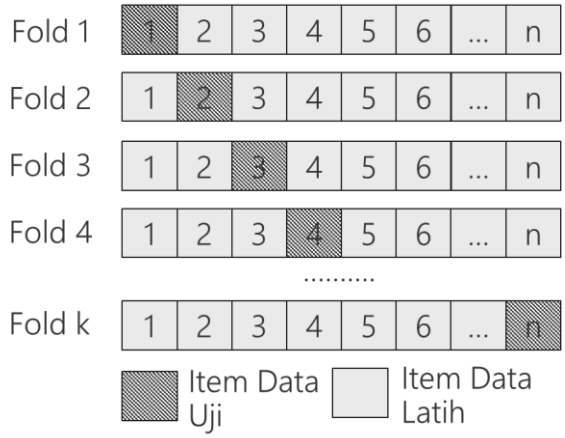
Cross-validation (CV) adalah metode statistik yang dapat digunakan untuk mengevaluasi kinerja model atau algoritma dimana data dipisahkan menjadi dua subset yaitu data latih dan data uji. Model atau algoritma dilatih oleh subset pembelajaran dan divalidasi oleh subset validasi. Selanjutnya pemilihan jenis CV dapat didasarkan pada ukuran dataset.

Untuk mengurangi variabilitas, maka dilakukan beberapa putaran validasi silang dengan subset berbeda dari data yang sama. Lalu hasil evaluasi tersebut digabungkan (rata-rata). Metode CV akan memberi kami perkiraan performa model yang lebih akurat dibandingkan Holdout.

Berikut ini beberapa jenis CV diantaranya :

1. LOOCV (*Leave one out cross-validation*)

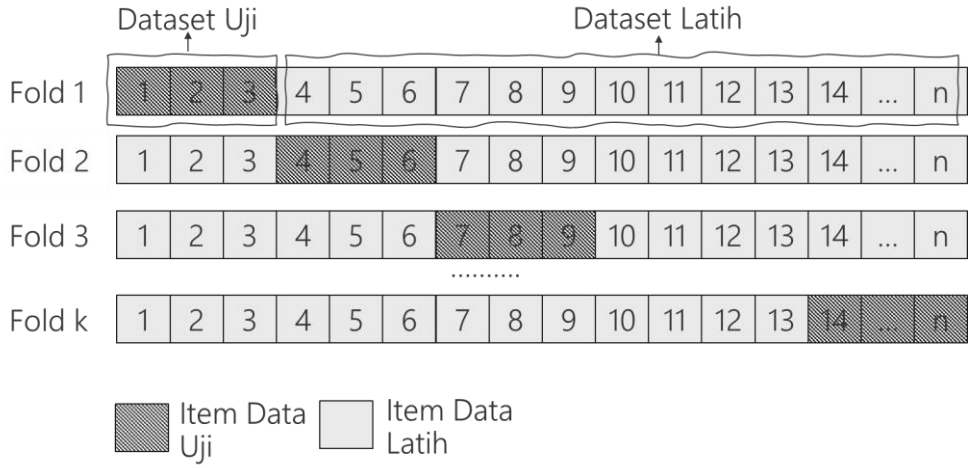
Pada LOOCV, data dibagi menjadi dua bagian, yaitu data uji dan data latih, namun pada data latih hanya menggunakan 1 data. Sehingga jika kita memiliki ***n*** data, maka kita akan memiliki sejumlah ***n-1*** data latih dan 1 data uji. Proses *training* dan evaluasi dilakukan sebanyak *n* kali. Ilustrasi LOCV dapat dilihat pada Gambar 3.11



Gambar 3.11. Ilustrasi *Leave one out cross-validation*

2. K-Fold cross-validation

K-fold adalah sebuah metode yang memecah dataset menjadi dua bagian yaitu data latih dan data uji sebanyak k kelompok dimana jumlah data latih dan data uji pada tiap-tiap kelompok sama. Sehingga kita akan melakukan perulangan sebanyak K kali untk melakukan training dan evaluasi. K dapat berisi sebuah nilai bulat. Jika k bernilai 1 maka metode ini sama dengan holdout. Biasnya nilai k lebih besar dari 1. Ilustrasi K-Fold ditunjukkan pada Gambar 3.12 .

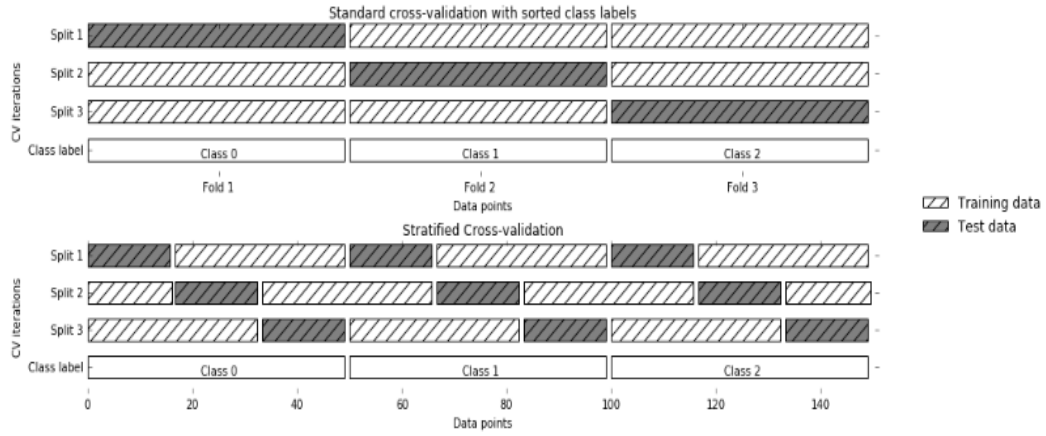


Gambar 3.12. Ilustrasi *K-Fold cross-validation*

10 fold CV adalah salah satu K-fold CV yang direkomendasikan untuk pemilihan model terbaik karena cenderung memberikan estimasi akurasi yang kurang bias dibandingkan dengan CV biasa dan leave-one-out CV. Dalam 10 fold CV, data dibagi menjadi 10 fold berukuran kira-kira sama, sehingga kita memiliki 10 subset data untuk mengevaluasi kinerja model atau algoritma. Untuk masing-masing dari 10 subset data tersebut, CV akan menggunakan 9 fold untuk pelatihan dan 1 fold untuk pengujian.

3. Stratified cross-validation

Stratified cross-validation mirip dengan K-Fold, namun perbedaannya dalam pembagian data tiap iterasinya memperhatikan kondisi data set seperti distribusi kelas, rata-rata dan varian, sehingga pembagian distribusi kelas lebih merata. Perhatikan Gambar 3.13 sebagai ilustrasi, dimana pada gambar tersebut distribusi kelas M dan L hampir sama pada tiap bagian data.



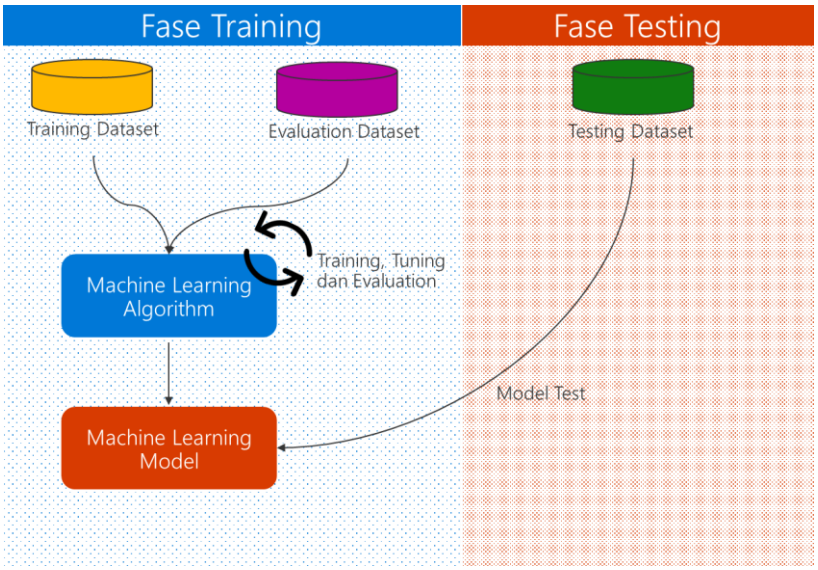
Gambar 3.13. Ilustrasi *Stratified cross-validation*

3.3 Pelatihan dan Uji Coba Model

Proses pelatihan model (*model training*) adalah sebuah proses mengekstrak pola atau pengetahuan pada data latih menggunakan algoritma tertentu. Berbagai jenis algoritma dapat digunakan pada tahap ini tergantung *task* atau masalah apa yang ingin dipecahkan. Hasil atau luaran

pelatihan menggunakan algoritma tertentu adalah adalah sebuah model *Machine Learning.*

Pada proses fase pelatihan, digunakan dua jenis dataset yaitu *training* dan *validation dataset*. Algoritma ML memanfaatkan *training* dataset yang telah disiapkan pada fase sebelumnya untuk mengekstrak pola atau pengetahuan pada data. Sedangkan Validation dataset, seting digunakan untuk memvalidasi model dan melakukan *hyperparameter optimalization.*

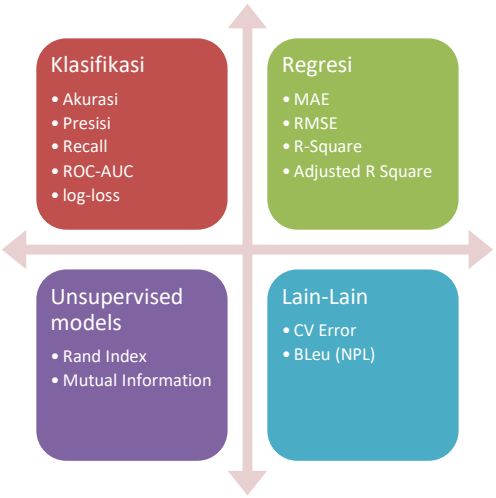


Gambar 3.14. Proses Ilustrasi Pelatihan dan Pengujian Model

Pada tahap training juga biasanya dilakukan *hyperparameter optimalization* (Tuning) dan *evaluation*. *Hyperparameter* adalah parameter yang harus ditetap sebelum proses pembelajaran berlangsung. *Hyperparameter* ini berbeda-beda tiap algoritma dan nilainya mempengaruhi performa dan dipengaruhi oleh masalah dan data yang ingin dipecahkan. Oleh karena itu, biasanya pada proses ini dilakukan berulang-ulang sehingga diperoleh beberapa model lalu dipilih model yang terbaik. Untuk detail proses ini, anda dapat merujuk pada bab selanjutnya tergantung algoritma apa yang akan digunakan.

3.3.1 Evaluasi model

Evaluasi model adalah sebuah proses untuk mengukur performa model yang telah dihasilkan pada proses *training*. Ada banyak metode untuk mengevaluasi model, biasanya tergantung kepada jenis *task* yang dilakukan serta informasi apa yang ingin diketahui. Sebagai contoh pada kasus regresi dan klasifikasi, membutuhkan metode yang berbeda karena ukuran baik atau tidaknya model regresi dan klasifikasi berbeda.

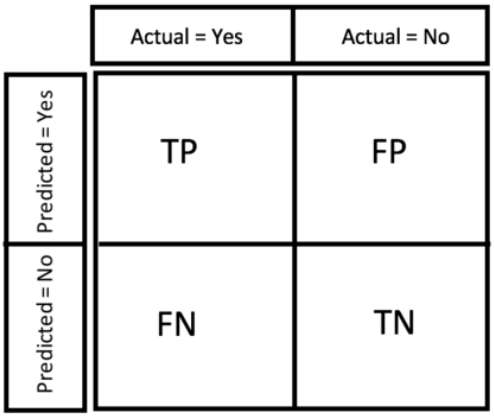


Gambar 3.15. Jenis-Jenis Metrik Evaluasi Model berdasarkan *task*

Adapun beberapa metode atau metric yang umum digunakan adalah

1. Confusion Matrix

Confusion matrik adalah matrik yang berukutan N x N dimana N adalah jumlah kelas yang diprediksi. Jadi metric ini cocok digunakan untuk permasalahan klasifikasi. *Confusion matrix* menyajikan ringkasan semua hasil prediksi yang dihasilkan dengan membandingkan antara hasil prediksi dan hasil yang diharapkan.

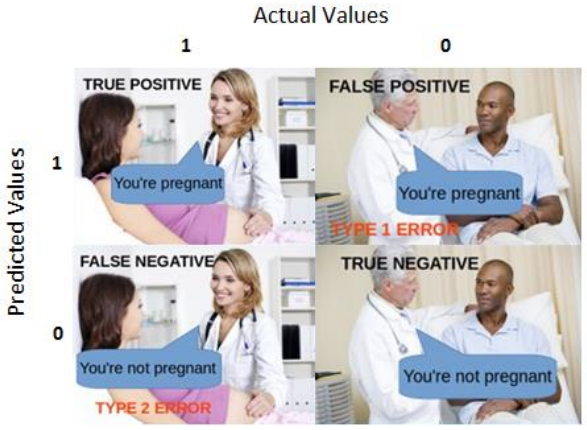


Gambar 3.16. Contoh confusion matrix dengan dua kelas yaitu yes dan no

Sebagai contoh perhatikan Gambar 3.16 yang menggambarkan sebuah *Confusion matrix* untuk dua kelas yaitu Yes dan No. Pada kolom baris paling atas **Actual=Yes** berarti pada real data atau label bernilai Yes dan **Actual=No** berarti pada kenyataan atau real data bernilai No. Sedangkan pada kolom **Predicted=Yes** berarti nilai prediksi dari model bernilai Yes dan seterusnya kolom **Predicted=No** berarti nilai prediksi dari model bernilai No. Pada confusion matrik tersebut ada 4 kolom yaitu

- TP (*True Positif*) berisi jumlah data points diberi label Yes yang memang sebenarnya bernilai Yes.
- TN (*True Negatif*) berisi jumlah data points diberi label No yang memang sebenarnya bernilai No.
- FP (*False Positif*) berisi jumlah data points diberi label Yes yang memang sebenarnya bernilai No. *Error* ini biasa disebut *Error Type 1*.
- FN (*False Negatif*) berisi jumlah data points diberi label No yang memang sebenarnya bernilai Yes. *Error* ini biasa disebut *Error Type 2*.

Meme berikut ini menggambarkan *Confusion matrix* dengan jelas, dimana tiap kolomnya merepresentasikan keputusan dokter tentang seseorang hamil atau tidak



Gambar 3.17. Contoh confusion matrix dengan dua kelas yaitu yes dan no

Berdasarkan nilai-nilai *Confusion matrix* tersebut maka dapat dihitung metric-metric berikut

- **Akurasi :** Perbandingan jumlah item yang di prediksi benar dengan total seluruh prediksi yang dilakukan.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Recall atau Sensitivity atau TPR (*True Positive Rate*)**: Perbandingan jumlah item yang relevan diidentifikasi benar dengan seluruh item yang benar. Contohnya seberapa besar orang yang positif Covid di identifikasi sebagai positif ?

$$Recall = \frac{TP}{TP + FN}$$

- **Specificity atau TNR (*True Negative Rate*):** Perbandingan jumlah item yang di identifikasi sebagai negatif dengan yang benar-benar negatif. Contohnya berapa banyak orang yang sehat di identifikasi tidak terkena Covid? Adapun Specificity di ukur dengan mengitung :

$$Specificity = \frac{TN}{TN + FP}$$

- **Precision:** Perbandingan jumlah item yang di identifikasi sebagai positif secara benar terhadap jumlah item yang di identifikasi positif.

$$Precision = \frac{TP}{TP + FP}$$

- ***False Positive Rate* (FPR) atau *Type I Error*:** Perbandingan jumlah item yang salah identifikasi sebagai positif dengan jumlah item yang dibenar-benar negatif.

$$FPR = \frac{FP}{FP + TN} = 1 - Specifity$$

- ***False Negative Rate* (FNR) atau *Type II Error:*** Perbandingan jumlah item yang salah identifikasi sebagai negatif dengan jumlah item yang dibenar-benar positif. FNR = FN/(FN+TP)

$$FNR = \frac{FN}{FN + TP}$$

Contoh Soal

Ukurlah kinerja dari sebuah mesin pemisah buah apel berdasarkan warna yaitu apel merah dan apel hijau. Dalam proses pengujian digunakan 100 Apel Merah dan 900 Apel Hijau. Hasilnya mesin tersebut memisahkan 110 yang dideteksi sebagai Apel Merah. Ke 110 apel tersebut kemudian dicek kembali oleh manusia, ternyata dari 110 apel tersebut hanya 90 apel yang merupakan Apel Merah, sedangkan 20 lainnya merupakan Apel Hijau.

Berdasarkan data maka dapat dibentuk Confusion matrik sebagai berikut:

	Realita:Merah	Realita: Hijau	
Prediksi: Merah	TP 90	FP 20	110 (Total item yang diprediksi merah)
Prediksi: Hijau	TN 10	FN 880	890 (Total item yang diprediksi Hijau)
	100 (Total item yang merah)	900 (Total item yang Hijau)	

Berdasarkan Confusion matrix ini dapat dihitung

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} = \frac{90 + 880}{90 + 10 + 20 + 880} =$$

$$Recall = \frac{TP}{TP + FN} = \frac{90}{90 + 880} =$$

$$Specificity = \frac{TN}{TN + FP} = \frac{10}{10 + 20} =$$

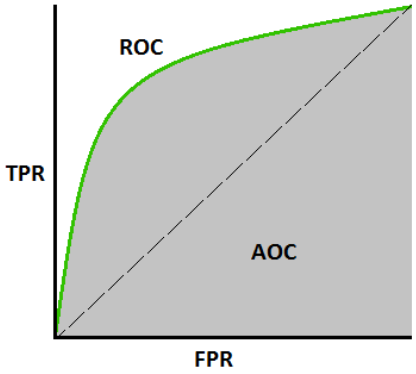
$$Precision = \frac{TP}{TP + FP} = \frac{90}{90 + 20} =$$

$$FPR = \frac{FP}{FP + TN} = \frac{20}{20 + 10} =$$

2. ROC dan AUC

Kurva AUC - ROC merupakan pengukuran performa untuk masalah klasifikasi pada berbagai pengaturan *threshold*. ROC adalah kurva probabilitas dan AUC mewakili derajat atau ukuran keterpisahan. Ini memberi tahu seberapa besar model mampu membedakan antar kelas. Semakin tinggi AUC, semakin baik model memprediksi 0 sebagai 0 dan 1 sebagai 1. Dengan analogi, Semakin tinggi AUC, semakin baik model dalam membedakan antara pasien dengan penyakit dan tidak ada penyakit.

Kurva KOP diplotkan dengan TPR terhadap FPR dimana TPR berada pada sumbu y dan FPR pada sumbu x.



Gambar 3.18. Contoh grafik ROC

Model yang sangat baik memiliki AUC mendekati angka 1 yang berarti model tersebut memiliki ukuran keterpisahkan yang baik. Model yang buruk memiliki AUC mendekati 0 yang berarti model tersebut memiliki ukuran keterpisahan yang terburuk. Sebenarnya itu berarti membalas hasilnya. Ini memprediksi 0s sebagai 1s dan 1s sebagai 0s. Dan jika AUC 0,5, berarti model tidak memiliki kapasitas pemisahan kelas sama sekali.

3. RMSE

Root Mean Square Error (RMSE) adalah metode pengukuran dengan mengukur perbedaan nilai dari prediksi sebuah model dengan nilai observasi atau label. RMSE dihitung dari akar kuadrat *Mean Square Error*. Nilai RMSE dapat berkisar dari 0 hingga ∞. Keakuratan metode estimasi kesalahan pengukuran ditandai dengan adanya nilai RMSE yang kecil. Nilai RMSE rendah menunjukkan bahwa nilai yang dihasilkan oleh suatu model mendekati nilai observasinya. Sebuah model yang mempunyai nilai RMSE lebih kecil dikatakan lebih akurat daripada model yang memiliki nilai RMSE lebih besar. Adapun formula untuk mengitung adalah

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\mathrm{n}}}$$

Keterangan:

$\hat{y}$ = nilai hasil observasi

$\hat{y}$ = nilai hasil prediksi

i = urutan data pada dataset

n = jumlah data

4. R-Square

Koefisien determinasi (R Square atau R kuadrat) atau disimbolkan dengan "R2" yang bermakna sebagai sumbangan pengaruh yang diberikan variabel bebas atau variabel independent (X) terhadap variabel terikat atau variabel dependent (Y). R Square ini berguna untuk melihat seberapa besar kontribusi pengaruh yang diberikan variabel X secara simultan (bersama-sama) terhadap variabel Y. Nilai R squared berkisar antara 0 sampai 1 yang mengindikasikan besarnya kombinasi variabel independen secara bersama – sama mempengaruhi nilai variabel dependen. Semakin mendekati angka satu, model yang dikeluarkan oleh regresi tersebut akan semakin baik. Adapun formula penghitungan adalah

$$R2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - y_{avg})^2}$$

Sebagai contoh interpretasi nilai R Square, jika nilai R adalah sebesar 0,80 maka koefisien determinasi (R Square) adalah 0,80 x 0,80 = 0,64. Berarti kemampuan variabel bebas dalam menjelaskan varians dari variabel terikatnya adalah sebesar 64,0%. Berarti terdapat 36% (100%-64%) varians variabel terikat yang dijelaskan oleh faktor lain.

3.4 Model Deployment

Aktifitas model deployment merupakan aktivitas yang paling terakhir dilakukan. Model deployment ialah penerapan model-model *machine learning*, atau sederhananya, memasukkan model ke dalam mesin produksi, berarti membuat model yang tersedia untuk sistem bisnis yang lain.

Tantangan dalam men-deploy sebuah model adalah menjaga skalabilitas dari model. Ketika di area produksi variasi dan jumlah input yang akan di dapat model akan berbeda pada fase pelatihan. Olah karena itu menjaga performa model tetap baik adalah salah satu tujuan yang paling penting.

Sebagian besar model mengalami permasalahan *Concept Drift* yang mengakibatkan penurunan performa model. *Concept drift* sederhanaya adalah perbedaan distribusi pada data latih dan data yang diuji. Akibatnya, hal tersebut akan mengubah relasi antara *input* dan *output*. Oleh karena itu, Ketika model di-*deploy* maka model harus dimonitor. Semua model yang di deploy harus mencatat semua *input*, *output* dan eksepsi yang terjadi. Biasanya pengembang menyiapkan sebuah infrastruktur untuk manajemen log dan visualisasi performance model.

3.5 Studi Kasus 1. Pengumpulan data berita

Sebagai contoh kita ingin mengambil data berita dari detik.com. Data yang diambil adalah data berita yang berisi judul, isi berita dan tanggal. Kita tidak mungkin mencatat satu per satu item yang ada di detik.com. Kita dapat membuat sebuah robot yang akan mengumpulkan data yang kita inginkan.

Agar mempermudah, pada kasus ini penulis akan menggunakan Jupyter Notebook sebagai *tools* untuk melakukan *scrapping*. Adapun langkah-langkah yang dilakukan adalah :

3.5.1 Install library yang dibutuhkan

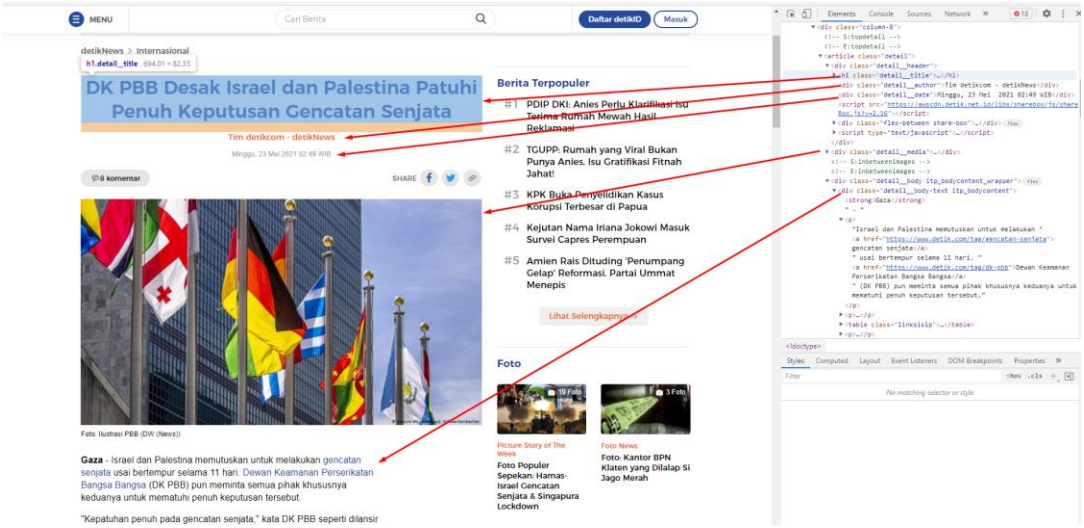
Untuk melakukan *scapping*, dibutuhkan beberapa *library* diantaranya Beautiful Soup dan Request. Untuk melakukan instalasi dapat menggunakan perintah berikut

```
[1]   1   !pip install BeautifulSoup4
      2   !pip install requests
```

Setelah library terinstall semua maka kita sudah suap melakukan scrapping.

3.5.2 Mengekstrak isi Halaman Web

Inspeksi bertujuan untuk mengetahui elemen apa yang harus dipanggil untuk melakukan ektraksi konten. Sebagai contoh satu halaman berita detik.com. Dengan bantuan tool inspect dari google chrome maka kita dapat menganlisa struktur halaman berita detik.com. Untuk mengetahui harus melihat source code atau menggunakan tool inspektor pada Gambar 3.19.



Gambar 3.19. Inspeksi terhadap elemen data detik.com

Hasil inspeksi menunjukkan bahwa:

- Judul berita terletak pada sebuah tag H1 dengan class detail__title
- Penulis terletak pada sebuah tag DIV dengan class detail__author
- Tanggal terletak pada sebuah tag DIV dengan class detail__data
- Isi berita terletak pada DIV dengan class detail__body-text, namun didalamnya ada iklan-iklan ke halaman lain dalam bentuk tabel, oleh karena itu ilkan ini harus dibersihkan

3.5.3 Parsing halaman webpage menggunakan Beautiful Soup

Setelah mengetahui struktur html maka langkah selanjutnya adalah membuat kode yang memparsing halaman web tersebut. Setelah melakukan request pada URL tertentu (baris 9), maka selanjutnya adalah menganalisa hasil request menggunakan Beautiful Soup (baris 10). Untuk mencari Judul maka perlu menemukan element berdasarkan tag H1 dengan class `detail__title` (baris 12), tanggal dan author pada baris 13 dan 14 . Selanjutnya adalah pembersihan konten dilakukan dengan menghapus semua elemen tabel pada teks.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np

#ambil berita detik
def getBeritaDetik(url):
  B = { }
  response = requests.get(url)
  soup = BeautifulSoup(response.text, 'html.parser')
  #ambil elemen-elemen berita
  B['judul'] = soup.find('h1', {'class': 'detail__title'}).text.replace('\n', "").strip()
  B['tanggal'] = soup.find('div', {'class': 'detail__date'}).text.replace('\n', "").strip()
  B['author'] = soup.find('div', {'class': 'detail__author'}).text.replace('\n', "").strip()
  berita = soup.find('div', {'class': 'detail__body-text'})
  text_berita = berita.text
  #bersihkan isi berita
  blah = berita.find_all("table")
  for x in blah:
    text_berita = text_berita.replace(x.text,'').replace('\n', "").strip()
    #print(x.text)
  B['berita'] = text_berita
  return(B)

```

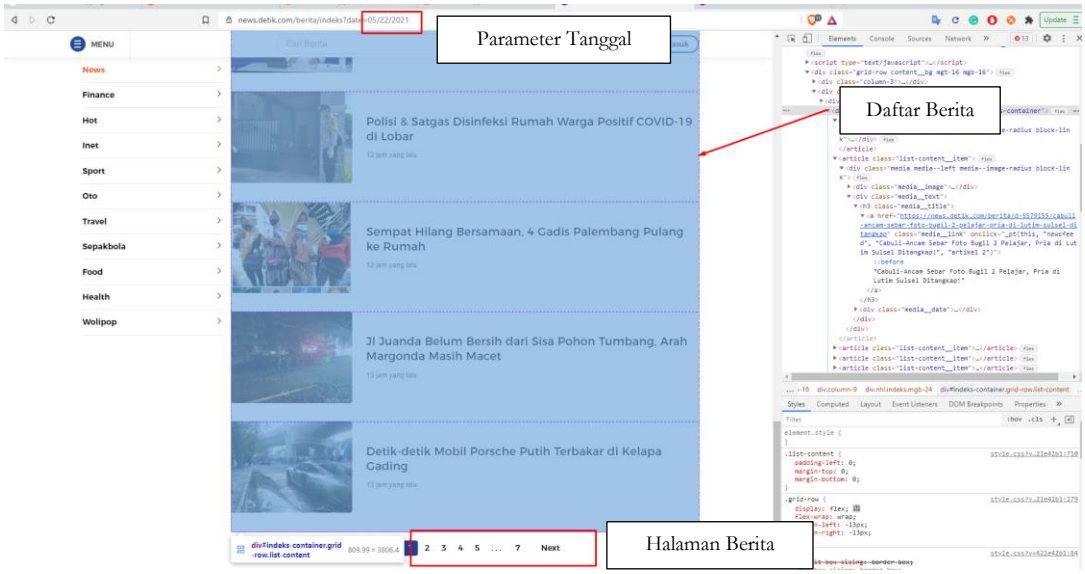
Method tersebut dapat digunakan untuk mengambil 1 halaman berita contohnya:

```
getBeritaDetik("https://news.detik.com/berita/"+
               "d-5579022/detik-detik-mobil-porsche-putih-terbakar-di-kelapa-gading")
```

```
{'author': 'Yogi Ernes - detikNews',
 'berita': 'Jakarta - Sebuah mobil sport merek Porsche hangus terbakar di Kelapa Gading, Jakarta Utara.
 'judul': 'Detik-detik Mobil Porsche Putih Terbakar di Kelapa Gading',
 'tanggal': 'Sabtu, 22 Mei  2021 19:49 WIB'}
```

3.5.4 Mengambil index daftar berita pertanggal

Setelah berhasil mengambil data per berita, maka langkah selanjutnya adalah mengambil seluruh berita. Index berita tersedia pada alamat https://news.detik.com/berita/indek dan dapat dilihat berdasarkan tanggal. Daftar berita pada tanggal tertentu dipisahkan berdasarkan halaman-halaman, karena jumlah berita pada tanggal tersebut cukup banyak. Hasil analisa menunjukkan bahwa kita dapat memberikan parameter date dan nomer halaman. Struktur halaman indek dpaat dilihat pada Gambar 3.20 Struktur Index berita detik.comGambar 3.20.



Gambar 3.20 Struktur Index berita detik.com

Semua konten terleetak pada div dengan kelas list-kontent dan judul dan url terletak pada tag article. Setelah mengambil list url yanga da pada artikel, maka berdasarkan link tersebut kita ambil detail beritanya (baris 21). Adapun source code untuk mengambil list adalah sebagai berikut

```
1  import requests
2  from bs4 import BeautifulSoup
3
4  def indexBerita (tanggal,jumlahHalaman):
5    daftarBerita = []
6    halaman=0
7    for halaman in range(0, jumlahHalaman):
8      halaman = halaman + 1
9      base_url = 'https://news.detik.com/berita/indeks/' + str(halaman)+"?date="+tanggal
10     #print(base_url)
11
12     # Request URL and Beautiful Parser
13     r = requests.get(base_url)
14     soup = BeautifulSoup(r.text, "html.parser")
15
16     berita_container = soup.find('div', {'id': 'indeks-container'})
17     berita = berita_container.find_all('article')
18
19     for item in berita:
20       x = item.find("a", href=True)
21       berita =getBeritaDetik(x['href'])
22       daftarBerita.append(berita)
23   return daftarBerita
24
```

Luaran dari method tersebut dapat kita oleh lebih lanjut atau disimpan melalui excel menggunakan pandas

```
tanggal ='05/22/2021'
jumlahHalaman=1
list_berita =indexBerita(tanggal,jumlahHalaman)
#olah pada dataframe
df = pd.DataFrame(list_berita)
print(df.columns)
print(df)
```

```
Index(['judul', 'tanggal', 'author', 'berita'], dtype='object')
                                                judul  ...                                             berita
0   Menhub: Masyarakat dari Sumatera ke Jawa Wajib...  ...  Jakarta - Menteri Perhubungan (Menhub) Budi Ka...
1   Cabuli-Ancam Sebar Foto Bugil 2 Pelajar, Pria ...  ...  Luwu Timur - Pemuda bernama Adrian (22) di Luw...
2   Mendes Minta Pendamping Desa Gotong-Royong Ban...  ...  Jakarta - Menteri Desa, Pembangunan Daerah Ter...
3         Gempa M 3,6 Terjadi di Waingapu Sumba Timur  ...  Jakarta - Gempa berkekuatan magnitudo (M) 3,6 ...
4   Beri Benih hingga Cold Storage, Mentan Harap P...  ...  Jakarta - Kementerian Pertanian (Kementan) ser...
5   MUI Setuju Saran JK soal Kotak Amal untuk Pale...  ...  Jakarta -  Majelis Ulama Indonesia (MUI) memin...
6   Dalih Ngantuk Pemobil Tabrak Lari Pedagang Mi ...  ...  Jakarta -  Berkat 'kesaktian' teknologi, penge...
7   Mendes Jadikan 2 Desa di Jatim Percontohan Pem...  ...  Jakarta - Menteri Desa, Pembangunan Daerah Ter...
8   KPK Bantah OTT Bupati Nganjuk Ditangani Baresk...  ...  Jakarta - Direktur Sosialisasi dan Kampanye An...
9   Jejak Dokter Asal Sumut Jual Vaksin Corona Ile...  ...  Jakarta -  Polda Sumatera Utara (Sumut) meneta...
10  Cegah Lonjakan COVID-19, Satgas Minta 7 Kota Z...  ...  Jakarta - Satgas Penanganan COVID-19 meminta p...
11  Libur Lebaran, Vaksinasi Lansia di NTB Tetap J...  ...  Jakarta - Vaksinasi lansia tetap dilaksanakan ...
12  Saling Serang Partai Ummat-Ngabalin Buntut Ami...  ...  Jakarta -  Tenaga Ahli Utama Kantor Staf Presi...
13  Tak Diizinkan, Puluhan Pesepeda Gagal Peringat...  ...  Jakarta - Beberapa organisasi kemasyarakatan s...
```

3.5.5 Mengubah dataset menjadi Excel

Untuk mengubah menjadi Excel maka cukup menjalankan perintah to_excel pada Pandas Dataframe anda. Contoh penggunaannya adalah berikut

```
df.to_excel("berita.xls")
```

	A	B	C	D	E	F
		judul	tanggal	author	berita	
0		Menhub: Masyarakat	Sabtu, 22 Mei 2021 23:44	Matius Alfons - detikNev	Jakarta - Menteri Perhubungan (Menhub) Budi Karya Sumadi meminta ag	
1		Cabuli-Ancam Sebar	Sabtu, 22 Mei 2021 23:13	Hermawan Mappiwali - d	Luwu Timur - Pemuda bernama Adrian (22) di Luwu Timur (Lutim), Sulawe	
2		Mendes Minta Penda	Sabtu, 22 Mei 2021 23:09	Jihaan Khoirunnisaa - de	Jakarta - Menteri Desa, Pembangunan Daerah Tertinggal dan Transmigras	
3		Gempa M 3,6 Terjadi	Sabtu, 22 Mei 2021 22:54	Tim detikcom - detikNev	Jakarta - Gempa berkekuatan magnitudo (M) 3,6 terjadi di Waingapu, Su	
4		Beri Benih hingga Co	Sabtu, 22 Mei 2021 22:48	Nadhifa Sarah Amalia - d	Jakarta - Kementerian Pertanian (Kementan) serahkan bantuan benih hor	
5		MUI Setuju Saran JK	Sabtu, 22 Mei 2021 22:40	Tim detikcom - detikNev	Jakarta - Majelis Ulama Indonesia (MUI) meminta agar Pemerintah Indon	
6		Dalih Ngantuk Pemol	Sabtu, 22 Mei 2021 22:23	Tim detikcom - detikNev	Jakarta - Berkat 'kesaktian' teknologi, pengemudi yang menabrak lari pe	
7		Mendes Jadikan 2 De	Sabtu, 22 Mei 2021 22:01	Inkana Putri - detikNews	Jakarta - Menteri Desa, Pembangunan Daerah Tertinggal dan Transmigras	
8		KPK Bantah OTT Bu	Sabtu, 22 Mei 2021 21:50	Matius Alfons - detikNev	Jakarta - Direktur Sosialisasi dan Kampanye Anti-Korupsi Komisi Pembe	
9		Jejak Dokter Asal Su	Sabtu, 22 Mei 2021 21:41	Tim detikcom - detikNev	Jakarta - Polda Sumatera Utara (Sumut) menetapkan empat orang, term	
10		Cegah Lonjakan COV	Sabtu, 22 Mei 2021 21:40	Erika Dyah - detikNews	Jakarta - Satgas Penanganan COVID-19 meminta pemerintah kabupaten/	
11		Libur Lebaran, Vaksir	Sabtu, 22 Mei 2021 21:28	Nadhifa Sarah Amalia - d	Jakarta - Vaksinasi lansia tetap dilaksanakan meski saat Idul Fitri. Vaksi	
12		Saling Serang Partai	Sabtu, 22 Mei 2021 21:01	Tim detikcom - detikNev	Jakarta - Tenaga Ahli Utama Kantor Staf Presiden (KSP) Ali Mochtar Nga	
13		Tak Diizinkan, Puluh	Sabtu, 22 Mei 2021 20:59	Adhyasta Dirgantara - d	Jakarta - Beberapa organisasi kemasyarakatan sipil bersama mahasiswa	
14		Kisah Ipda Tita, Polw	Sabtu, 22 Mei 2021 20:30	Yogi Ernes - detikNews	Serpong - Kasus penganiayaan anak yang dilakukan ayahnya sendiri di S	
15		Tekan COVID-19, Sa	Sabtu, 22 Mei 2021 20:30	Inkana Putri - detikNews	Jakarta - Juru Bicara Satgas Penanganan COVID-19 Prof Wiku Adisasmi	
16		Polisi & Satgas Disin	Sabtu, 22 Mei 2021 20:28	Erika Dyah - detikNews	Jakarta - Bhabinkamtibmas Desa Midang Polsek Gunungsari, Aipda Dew	
17		Sempat Hilang Bersa	Sabtu, 22 Mei 2021 20:16	Prima Syahbana - detikI	Palembang - Empat gadis di Palembang, Sumatera Selatan, yang dilapo	
18		Jl Juanda Belum Ber	Sabtu, 22 Mei 2021 20:10	Adhyasta Dirgantara - d	Depok - Sejumlah pohon tumbang di beberapa titik di Jl Ir H Juanda, Dep	
19		Detik-detik Mobil Por	Sabtu, 22 Mei 2021 19:49	Yogi Ernes - detikNews	Jakarta - Sebuah mobil sport merek Porsche hangus terbakar di Kelapa G	

Gambar 3.21 Luaran Berita Detik.com tanggal 22 Mei 2021

3.6 Studi Kasus - *Data Preprocessing*

Untuk mempermudah, melakukan *data preprocessing*, penulis menyarankan untuk menggunakan Jupyter Notebook karena dapat mempermudah dalam menganalisa dan memperbaiki data. Adapan tahapan-tahapan dalam *preprocessing* dapat berbeda-beda, tergantung kepada permasalahan yang ada. Pada kasus ini penulis akan menganalisa dataset Titanic.

Adapun langkah-langkah yang dilakukan adalah:

3.6.1 Informasi Dataset

Dataset yang digunakan adalah *dataset* Titanic. Dataset ini berisi sejumlah daftar penumpang kapal Titanic yang menjadi korban dan selamat pada kecelakaan kapal tersebut. Dataset ini digunakan untuk memprediksi apakah seseorang selamat atau tidak berdasarkan data-data lainnya. Adapun informasi fitur pada data ini dapat dilihat pada Tabel 3.2

Tabel 3.2 Informasi fitur atau kolom dataset Titanic

Variable	Definisi	Penjelasan Isi
survival	Kolom label yang menyatakan penumpang selamat atau tidak	0 = Tidak selamat, 1 = Selamat
pclass	Kelas tiket penumpang	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Jenis kelamin penumpang	
Age	Umur penumpang dalam tahun	
sibsp	jumlah saudara kandung / pasangan di kapal Titanic	
parch	jumlah orang tua / anak di kapal Titanic	
ticket	Nomer tiket	
fare	Tarif Penumpang	
cabin	Nomor Kabin	

embarked	Pelabuhan keberangkatan	C = Cherbourg, Q = Queenstown, S = Southampton

Proses identifikasi kolom, tipe data dan isi dari sebuah dataset adalah langkah awal yang harus diketahui sebelum melakukan pemrosesan lebih lanjut.

3.6.2 Memuat dan menganalisa tipe data

Datase titanic terdiri atas dua dokumen yaitu train.csv dan test.csv. Train.csv berisi data yang akan digunakan untuk pelatihan dan test.csv adalah data yang akan digunakan untuk evaluasi. Untuk memuat data ke dalam notebook kita dapat menggunakan pandas dengan fungsi read_csv dengan parameter nama file. Pada kasus ini langkah tersebut dilakukan pada baris 11 dan 12. Setelah berhasil memuat data, kita tampilkan 5 data pertama dengan menggunakan fungsi head() (Baris 13)

```
1  # data analysis and wrangling
2  import pandas as pd
3  import numpy as np
4  import random as rnd
5
6  # visualization
7  import seaborn as sns
8  import matplotlib.pyplot as plt
9  %matplotlib inline
10
11 train_df = pd.read_csv('train.csv')
12 test_df = pd.read_csv('test.csv')
13 train_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Berdasarkan 5 data pertama dari fungsi head, kita telah dapat menentukan tipe-tipe data. Adapun tipe data yang ada pada dataset ini adalah

- Kategori : *Survived, Sex*, dan *Embarked*.
- Ordinal : Pclass
- Continous: *Age, Fare*.
- Discrete: SibSp, Parch.

Sedangkan untuk tipe data Ticket merupakan gabungan antara numerik dan alpha-numerik dan data Cabin adalah alpha-numerik dan dari analisa awal terdapat missing value (bernilai NaN). Dalam komputasi, NaN adalah singkatan dari *Not a Number*. NaN adalah nilai tipe data numerik yang mewakili nilai yang tidak ditentukan atau tidak terwakili.

```
[2]  1  train_df.info()
     2
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

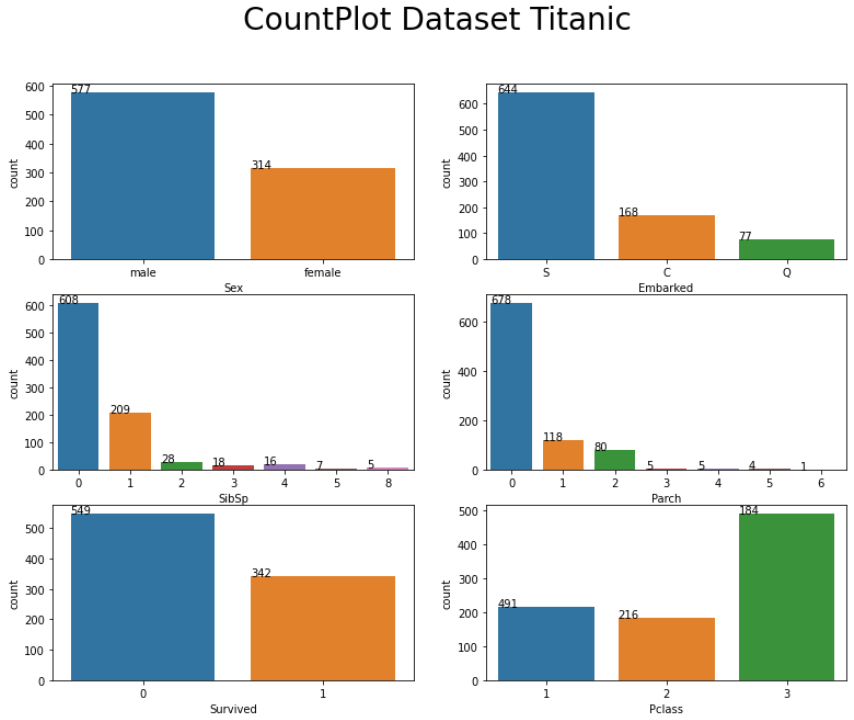
```
[3]   1   train_df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

3.6.3 Pengecekan Konsistensi Data

Berdasarkan analisa awal, kita memiliki tiga fitur yang bertipe kategori yaitu *Survived, Sex,* dan *Embarked,* dua discrete dan satu ordinal. Ketiga data tersebut akan dilihat sebaran nilainya dan konsistensi nilai (untuk kategori).

```
#pengaturan layout
n_rows=3
n_cols=2
#daftar kolom
kolom=["Sex","Embarked","SibSp","Parch","Survived","Pclass"]

fig, axes = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=(13, 10))
i=0
fig.suptitle('CountPlot Dataset Titanic', fontsize="28")
for row in range(n_rows):
    for col in range(n_cols):
      #menampilkan
      sns.countplot(x=kolom[i], data=train_df, ax=axes[row,col])
      #menampilkan label nilai per unique value
      for p, label in zip(axes[row,col].patches, train_df[kolom[i]].value_counts()):
        axes[row,col].annotate(label, (p.get_x(), p.get_height()))
      i=i+1


```



Pada fitur *sex* dan *embark* ini tidak ditemukan inkonsistensi data. Selanjutnya, pada fitur *Sex* terdapat dua nilai unik yakni *male* dan *female* dan kolom Embarked memiliki tiga nilai unik yakni S,C dan Q. Contoh inkonsistensi data pada *dataset* adalah apabila misalnya pada sex ditemukan nilai seperti male, M, Female, FEMALE. Data yang bernilai male dan M adalah sama tetapi memiliki value yang berbeda. Solusinya adalah dengan mengubah data-data yang berbeda tapi memiliki nilai yang sama seperi Female dan FEMALE menjadi nilai yang sama yaitu Female.

3.6.4 *Missing Data, Feature Selection* dan *Feature Scaling*

Pada dataset ini, kolom nama, tiket, Informasi kabin tidak digunakan sehingga kolom-kolom ini dapat dihapus

```
#Hapus Kolom yang tidak di inginkan.
train_df=train_df.drop(["Name","Ticket","Cabin","PassengerId"],axis=1)
train_df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	20.0	1	0	7.0	S
1	1	1	female	36.0	1	0	71.0	C
2	1	3	female	24.0	0	0	7.0	S
3	1	1	female	33.0	1	0	52.0	S
4	0	3	male	33.0	0	0	7.0	S

Selanjutnya, perubahan data kategorikal menjadi data numerik. Kita dapat menggunakan fungsi map pada dataframe. Pada kolom *sex, female* akan diubah menjadi 0 dan male menjadi 1. Begitu juga untuk Embarked S=0, C=1 dan Q=2

```
#KONVERT STRING VALUES(CATEGORICAL VALUES) MENJDI INTEGER
train_df.Sex=train_df.Sex.map({"female":0,"male":"1"})
train_df.Embarked=train_df.Embarked.map({"S":0,"C":"1","Q":2})
train_df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	20.0	1	0	7.0	0
1	1	1	0	36.0	1	0	71.0	1
2	1	3	0	24.0	0	0	7.0	0
3	1	1	0	33.0	1	0	52.0	0
4	0	3	1	33.0	0	0	7.0	0

Kolom *Age* masih memiliki *missing value* oleh karena itu kita dapat menggunakan median untuk mengisi umur yang kosong. Adapun cara mengubah nilai kosong menjadi nilai median adalah sebagai berikut

```
train_df['Age'] = train_df['Age'].fillna((train_df['Age'].median()))
```

Agar performa model bagus, kita dapat melakukan scaling terhadap dua nilai yaitu *age* dan *fare*

```
from sklearn.preprocessing import StandardScaler
train_df["Age"]=round((train_df.Age-train_df.Age.mean()/train_df.Age.std()))
train_df["Fare"]=round((train_df.Fare-train_df.Fare.mean()/train_df.Fare.std()))
```

Bab

4

K-Nearest Neighbors

Kemarahan dimulai dengan kegilaan dan berakhir dengan penyesalan. – Ali bin Abi Thalib.

Topik Pembelajaran:

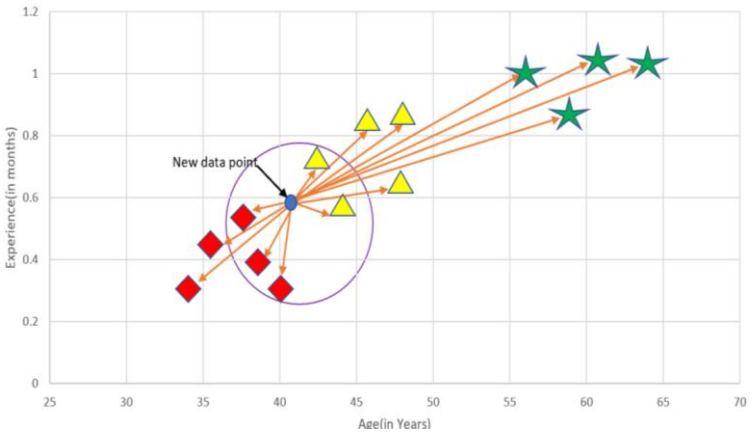
1. Definisi K-Nearest Neighboar
2. Bagaimana langkah-langkah Implementasi KNN
3. Keunggulan dan kelemahan KNN
4. Studi Kasus KNN

Nearest Neighbors adalah salah satu algoritma yang paling dasar dalam kasus klasifikasi. Algoritma ini biasa juga disebut K-Nearest Neighbors (KNN). KNN bersifat non parametric, yang artinya, ia tidak memerlukan asumsi distribusi sehingga sebaran data bebas dan merupakan jenis algoritma supervised learning yang bersifat *instance based*. Sederhananya, KNN tidak membuat generalisasi atau asumsi pada model, sehingga algoritma ini bebas untuk melakukan training pada segala jenis distribusi data. Algoritma ini biasanya digunakan untuk menyelesaikan permasalahan klasifikasi dan regresi.

KNN bekerja dengan cara membuat prediksi dengan menghitung kemiripan antara objek-objek yang berada di data training dengan objek yang ditest. Kemiripan dihitung dengan menggunakan fungsi "jarak".

4.1 Definisi K-Nearest Neighbors (KNN)

Algoritma KNN dapat didefinisikan berdasarkan namanya dimana K melambangkan sejumlah nilai tetangga terdekat (*Nearest Neighbors*) yang diguna untuk mengidentifikasi kemiripan sebuah titik baru dengan titik tetangganya. Sebagai contoh perhatikan Gambar 4.1, dimana kita ingin memiliki 3 kelompok data yaitu bintang, segitiga dan segiempat, kemudian kita diberikan sebuah titik baru yang tidak diketahui kelompoknya. Untuk menentukan kelompok titik baru tersebut menggunakan KNN dengan nilai K= 5 maka kita akan menghitung jarak antara titik yang baru dengan semua titik yang ada pada data, lalu mengambil 5 titik terdekat. Dari 5 titik terdekat itu dilakukan voting. Sehingga didapatkan hasil 2 titik adalah kelompok segitiga dan 3 titik kelompok segi empat. Maka dapat disimpulkan bahwa titik yang baru ini masuk kelompok titik segiempat. Bahasa sederhananya adalah kita melakukan voting berdasarkan tetangga terdekat untuk menentukan kelas atau kelompok sebuah titik baru.



Gambar 4.1. Ilustrasi KNN

Adapun langkah-langkah algoritma KNN adalah sebagai berikut:

1. Menentukan nilai K. Nilai K sebaiknya bilangan asli ganjil untuk menghindari hasil voting yang seimbang.
2. Menghitung jarak antara titik baru dengan semua data training.
3. Memilih K titik terdekat dengan titik yang baru

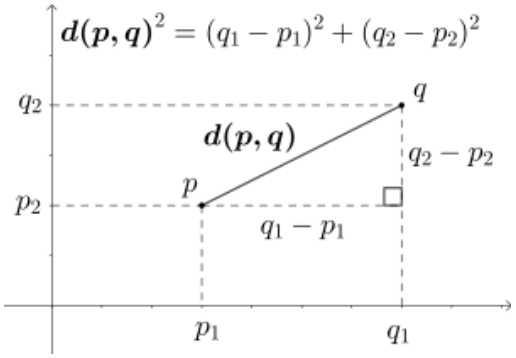
4. Pada kasus klasifikasi, dilakukan penghitungan jumlah titik terdekat tadi berdasarkan kategori/kelas. Titik yang baru akan masuk ke kategori yang memiliki jumlah titik terdekat terbanyak. Sedangkan pada kasus regresi, titik yang baru akan berada pada nilai rata-rata K tetangga terdekat.

4.2 Jarak Data KNN

Penghitungan jarak adalah kunci kesuksesan algoritma ini. Jarak melambangkan kemiripan antara data test dengan data training. Semakin jauh jarak berarti semakin tidak mirip, begitu juga sebaliknya. Ada beberapa cara yang dapat digunakan untuk menghitung jarak diantaranya:

4.2.1 Euclidean distance

Euclidean distance adalah perhitungan jarak dari 2 buah titik dalam Euclidean space. *Euclidean space* diperkenalkan oleh Euclid, seorang matematikawan dari Yunani sekitar tahun 300 B.C.E. untuk mempelajari hubungan antara sudut dan jarak. Euclidean ini berkaitan dengan Teorema Phytagoras dan biasanya diterapkan pada 1, 2 dan 3 dimensi. Tapi juga sederhana jika diterapkan pada dimensi yang lebih tinggi. Sederhananya jarak ini sering dikatakan sebagai jarak garis lurus antara dua titik.



Gambar 4.2. Ilustrasi *Euclidean distance*

Sebagai contoh, misalnya diketahui usia dan berat budi adalah 25 tahun dan 65 kg, andi berusia 27 tahun dengan berat badan 70 kg. Pertanyaan yang

ingin dijawab adalah siapa yang mirip dengan umar yang berusia 26 tahun dengan berat badan 67 kg?

Pertanyaan itu bisa dijawab menggunakan pendekatan jarak euclidean. Anggap saja variabel usia adalah x dan variabel berat badan adalah y. Maka, implementasinya seperti ini:

d (budi, umar) = $\sqrt{(25-26)^2 + (65-67)^2} = \sqrt{1+4} = 2.24$

d(andi, umar) = $\sqrt{(27-26)^2 + (70-67)^2} = \sqrt{1+9} = 3.16$

Kesimpulannya, saya mirip umar? anda sudah tahu jawabannya. Mana jarak yang paling dekat diantara 2,24 dan 3,16. Tentu saja 2,24. Jadi, semakin kecil nilai jarak eulidean maka semakin mirip. Jika hasilnya 0? berarti bukan hanya mirip tapi sama.

Formula umum dari jarak euclid antara titik p dan q adalah sebagai berikut dimana n menunjukkan dimensi data

$$d(p,q) = \sqrt{(p_1-q_1)^2 + \cdots + (p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2} \qquad (4\text{-}1)$$

Adapun implementasi dari persamaan diatas dapat dilihat pada *source code* berikut ini dimana kita akan mencari jarak diantara dua vektor.

```
from math import sqrt

# menghitung calculate euclidean
def euclidean_distance(a, b):
  return sqrt(sum((e1-e2)**2 for e1, e2 in zip(a,b)))

# define data
d1 = [12, 8, 9, 0, 0, 1]
d2 = [20, 10, 2, 1, 1, 0]
# calculate distance
dist = euclidean_distance(d1, d2)
print(dist)
```

```
10.954451150103322
```

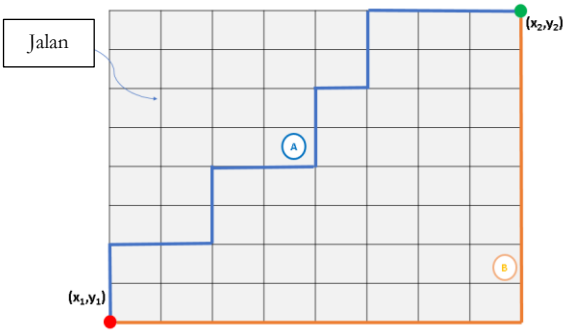
4.2.2 *Manhattan distance*

Manhattan distance atau Jarak Manhattan antara dua titik adalah jumlah dari panjang ruas garis kedua titik tersebut terhadap tiap sumbu dalam koordinat Kartesius. Jarak ini disebut juga dengan panjang Manhattan, jarak taksi, atau jarak *snake*. Nama jarak ini berasal dari tata letak jalan di pulau Manhattan yang berbentuk kisi-kisi segi empat. Jarak ini telah digunakan dalam analisis regresi sejak abad ke-18, dan saat ini umum dirujuk dengan LASSO.

Jarak Manhattan dalam ruang R dengan sistem koordinat Kartesius, antara vektor p = {p_1,p_2,p_3,..,p_n} dan q={q_1,q_2,q_3,..,q_n} adalah jumlah panjang proyeksi ruas garis antara kedua vektor tersebut terhadap sumbu-sumbu koordinat. Secara matematis, jarak Manhattan dapat didefinisikan sebagai berikut

$$d(p,q) = |p_1 - q_1| + |p_2 - q_2| + \cdots + |p_n - q_n| = \sum_{i=1}^{n} |p_i - q_i| \qquad (4\text{-}2)$$

Ilustrasi dari pengukuran jarak ini dapat dilihat dari gambar4.3 dimana untuk pergi dari titik (x1,y1) ke titik (x2,y2) melalui jalan yang berbentuk blok maka akan terdapat banyak kombinasi jalan. Perhatikan jalan A dan jalan B. Walaupun bentuknya berbeda namun total jarak tempuhnya sama.



Gambar 4.3. Ilustrasi *Manhattan distance*

Adapun implementasi dari persamaan diatas dapat dilihat pada *source code* dibawah ini

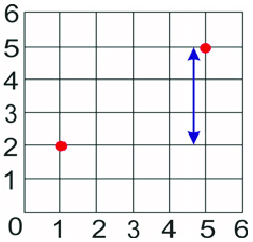
```python
from math import sqrt

# menghitung manhattan distance
def manhattan_distance(a, b):
  return sum(abs(e1-e2) for e1, e2 in zip(a,b))

# define data
d1 = [12, 8, 9, 0, 0, 1]
d2 = [20, 10, 2, 1, 1, 0]
# calculate distance
dist = manhattan_distance(d1,d2)
print(dist)
```

```
20
```

Jarak Manhattan berfungsi dengan baik untuk data berdimensi tinggi, namun metrik ini adalah ukuran yang kurang intuitif dibandingkan jarak euclidean, terutama bila digunakan dalam data dimensi tinggi. Ketika digunakan pada data diskrit dan/atau biner, Manhattan distance memperhitungkan jalur yang secara realistis dapat diambil dalam nilai atribut tersebut. Selain itu, jarak tersebut lebih cenderung memberikan nilai jarak yang lebih tinggi daripada jarak euclidean karena bukan merupakan jalur terpendek yang antara dua titik.

4.2.3 Chebyshev Distance

Chebyshev Distance atau Jarak Chebyshev didefinisikan sebagai selisih terbesar antara dua vektor sepanjang dimensi koordinat mana pun. Dengan kata lain, ini hanyalah jarak maksimum sepanjang satu sumbu. Karena sifatnya, itu sering disebut sebagai jarak Papan Catur karena jumlah minimum gerakan yang dibutuhkan oleh raja untuk berpindah dari satu kotak ke kotak lainnya sama dengan jarak Chebyshev.



Gambar 4.4. Ilustrasi *Chebyshev distance*

Chebyshev biasanya digunakan dalam kasus penggunaan yang sangat spesifik, yang membuatnya sulit untuk digunakan sebagai metrik jarak serba guna, seperti jarak Euclidean atau *Cosine distance*. Oleh karena itu, disarankan untuk hanya menggunakannya jika Anda benar-benar yakin metode ini sesuai dengan kasus penggunaan Anda.

4.2.4 Minkowski Distance

Jarak Minkowski atau metrik Minkowski adalah metrik dalam ruang vektor yang dapat disebut sebagai generalisasi jarak Chebyshev, Euclid dan jarak Manhattan. Jarak ini dinamai dari Hermann Minkowski, matematikawan asal Jerman. Adapun formula dari jarak ini adalah

$$d(p,q) = \sqrt[p]{\left(\sum_{i=1}^{n} |p_i - q_i|^p\right)} \quad (4\text{-}3)$$

Ketika menggunakan jarak ini, kita harus menentukan parameter P. Tantangannya adalah memahami berapa nilai P yang sesuai untuk kasus yang dihadapi. Berikut ini Nilai-nilai umum p adalah:

- p = 1 untuk Jarak Manhattan
- p = 2 untuk Jarak Euclidean
- p = ∞ untuk Jarak Chebyshev, dengan menemukan limit dari vector tersebut maka sebenarnya jarak ini adalah nilai maksimum dari vector-vektor tersebut.

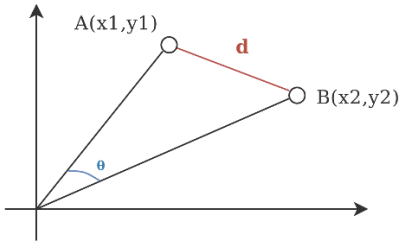
```
from math import sqrt

# menghitung minkowski distance
def minkowski_distance(a, b, p):
  return sum(abs(e1-e2)**p for e1, e2 in zip(a,b))**(1/p)

# define data
d1 = [12, 8, 9, 0, 0, 1]
d2 = [20, 10, 2, 1, 1, 0]
```

```
# menghitung distance (p=1)
dist = minkowski_distance(d1,d2, 1)
print("P1: ",dist)
# menghitung distance (p=2)
dist = minkowski_distance(d1,d2, 2)
print("P2: ",dist)
```

```
P1:  20.0
P2:  10.954451150103322
```

4.2.5 Cosine Distance

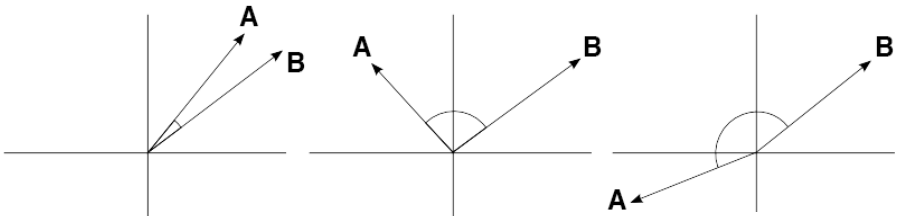
Sebelum memahami *Cosine Distance*, sebaiknya mengenali *Cosine similarity*. *Cosine similarity* atau kesamaan kosinus sederhanya adalah kosinus dari sudut (θ) antara dua vektor. Pada gambar 4.5, terdapat dua vector A dan B. d merupakan jarak Euclid dari kedua vector ini dan cos θ menunjukaan seberapa sama kedua vector tersebut. Semakin kecil sudut θ, maka semakin mirip kedua vector dan semakin dekat jaraknya. Jarak ini merupakan alternatif dari jarak Euclidean pada data dengan dimensi tinggi.



Gambar 4.5. Ilustrasi *Cosine distance*

Sebagaimana yang kita ketahui, nilai cos berada pada jarak -1 sampai 1. Oleh karena itu maka ada beberapa kemungkinan yang terjadi diantaranya:

- Ketika similarity(A,B) = 0, dimana θ = 90° berarti kesamaan antara vector tersebut sangat kecil bahkan dapat dikatakan tidak berelasi.
- Ketika similarity(A,B) = -1, dimana θ = 180° berarti vektor tersebut sangat berbeda atau berlawanan.
- Ketika similarity(A,B) = 1, dimana θ = 0° berarti vektor tersebut sama atau sangat mirip.



Vektor A dan B Memiliki Kesamaan | Vektor A dan B Tidak Berelasi | Vektor A dan B Berlawanan

Salah satu kelemahan utama dari kesamaan kosinus adalah besaran vektor tidak diperhitungkan, hanya arahnya. Dalam praktiknya, perbedaan nilai tersebut tidak sepenuhnya diperhitungkan. Jika Anda mengambil sistem pemberi rekomendasi, misalnya, kesamaan kosinus tidak memperhitungkan perbedaan dalam skala penilaian antara pengguna yang berbeda.

Persamaan *cosine similarity* dan hubungannya dengan *cosine distance* ditunjukkan oleh persamaan berikut

$$CosineSimilarity(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}}$$

$$CosineDistance\ (A,B) = 1\ -\ CosineSimilarity(A,B)$$

Cosine distance cocok digunakan pada vektor berdimensi tinggi dan besaran vektor tidak penting. Untuk analisis teks, ukuran ini cukup sering digunakan ketika data diwakili oleh jumlah kata. Misalnya, ketika sebuah kata lebih sering muncul dalam satu dokumen daripada dokumen lainnya, ini tidak berarti bahwa satu dokumen lebih terkait dengan kata itu. Bisa jadi dokumen memiliki panjang yang tidak rata dan besarnya hitungan kurang penting.

Berikut ini adalah implementasi dari *Cosine distance* dan *cosine similarity* pada data list 1 dimensi.

```python
#implementasi formula cosine halaman 141
def CosineSimilarity(A, B):
  #hitung penjumlahan vektor A*B
  dot = sum(a*b for a, b in zip(A, B))
  norm_a = sum(a*a for a in A) ** 0.5
  norm_b = sum(b*b for b in B) ** 0.5
  return dot / (norm_a*norm_b)

def CosineDistance (A, B):
  return 1- CosineSimilarity(A,B)

# uji coba
vec_x = [1,2,4,5,7,8]
vec_y = [1,2,4,5,7,6]

print('Similarity:', CosineSimilarity(vec_x,vec_y))
print('Distance:', CosineDistance(vec_x,vec_y))

```

```
Similarity: 0.9908361142442783
Distance: 0.00916388575572169
```

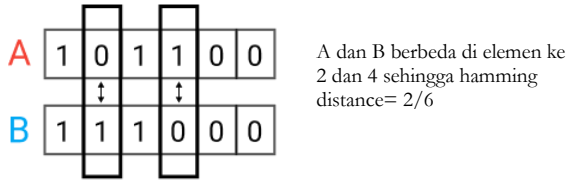
Dari contoh diatas terlihat bahwa vec_x dan vect_y hanya berbeda 1 item dan memiliki kesamaan 0.99 dan jarak 0.01. implementasi diatas hanya bisa digunakan untuk vektor, jika inging menggunakan data matriks atau yang lebih kompleks dapat menggunakan library Scikit-learn seperti contoh berikut ini

```python
from sklearn.metrics.pairwise import cosine_similarity

print('Similarity:', cosine_similarity([vec_x],[vec_y]))
```

```
Similarity: [[0.99083611]]
```

4.2.6 Hamming Distance

Jarak Hamming (*Hamming Distance*) adalah banyaknya nilai yang berbeda antara dua vektor. Ini biasanya digunakan untuk membandingkan dua string biner dengan panjang yang sama. Ini juga dapat digunakan untuk string untuk membandingkan seberapa mirip satu sama lain dengan menghitung jumlah karakter yang berbeda satu sama lain.



Gambar 4.6. Ilustrasi *Hamming distance*

Kelemahan jarak hamming sulit digunakan jika dua vektor tidak memiliki panjang yang sama. Anda ingin membandingkan vektor dengan panjang yang sama satu sama lain untuk memahami posisi mana yang tidak cocok.

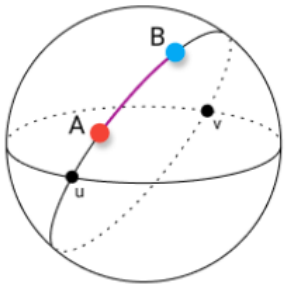
Selain itu, nilai sebenarnya tidak diperhitungkan selama nilainya berbeda atau sama. Oleh karena itu, tidak disarankan untuk menggunakan pengukur jarak ini jika besarannya adalah ukuran yang penting. Berikut ini contoh implementasi dari jarak hamming pada data list

```
def hamming_distance(A, B):
  jarak = 0
  for n in range(len(A)):
    #jika item posisi N sama
    if A[n] != B[n]:
      jarak += 1
  #dibagi dengan panjang A, agar nilainya
  #antara 0-1
  return jarak/len(A)

vec_x = [1,2,4,5,7,8]
vec_y = [1,2,4,5,2,6]
print(hamming_distance(vec_x,vec_y))
```

```
0.3333333333333333
```

4.2.7 Haversine Distance

Jarak Haversine adalah jarak antara dua titik pada sebuah bola berdasarkan bujur dan lintangnya. Ini sangat mirip dengan jarak Euclidean yang menghitung garis terpendek antara dua titik. Perbedaan utamanya adalah tidak ada garis lurus yang memungkinkan karena asumsi di sini adalah bahwa kedua titik tersebut berada pada sebuah bola.



Gambar 4.7. Ilustrasi *Haversine distance*

Salah satu kelemahan dari alat ukur jarak ini adalah diasumsikan bahwa titik-titik tersebut terletak pada sebuah bola . Dalam praktiknya, hal ini jarang terjadi karena, misalnya, bumi tidak bulat sempurna yang dapat mempersulit perhitungan dalam kasus-kasus tertentu.

Jarak Haversine sering digunakan dalam navigasi dan penghitungan jarak pada sistem geografi. Misalnya, Anda dapat menggunakannya untuk menghitung jarak antara dua negara. Pada jarak dekat, lengkungan tidak akan berdampak besar, berbeda untuk jarak yang jauh.

Haversine dapat dihitung menggunakan formula berikut. r adalah radius bola, dalam kasus ini adalah 6371 KM atau 3956 Miles (gunakan sesuai satuan yang anda suka). Input dari formula ini adalah latitude (φ) dan longitude (λ).

$$
\begin{aligned}
d &= 2r \arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)}\right) \\
&= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)
\end{aligned}
$$

Berikut ini adalah implementasi dari jarak haversine mengunakan koordinat latitude dan longitude. Pada contoh berikut diketahui titik koordinat dua kota yaitu pekanbaru dan jakarta.

```python
from math import radians, cos, sin, asin, sqrt

def haversine(lon1, lat1, lon2, lat2):

    # konversi degree to radian
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # radius bumi
    return c * r

#kota pekanbaru
lat1 = 0.510440
lon1 = 101.438309
#jakarta
lat2 = -6.200000
lon2 = 106.816666

print("Jarak Pekanbaru Jakarta : ",haversine(lat1,lon1,lat1,lon2)," KM")
```

```
Jarak Pekanbaru Jakarta :  598.0460120832477  KM
```

4.3 Implementasi K-NN

Implementasi KNN sangatlah sederhana. Proses pelatihan pada KNN hanya memasukkan atau menyimpan semua data latih berserta labelnya kedalam sebuah variabel yang akan digunakan untuk evaluasi (baris 13 dan 14).

```
1  import numpy as np
2  from collections import Counter
3
4  def euclidean_distance(x1, x2):
5    return np.sqrt(np.sum((x1 - x2)**2))
6
7  class KNN:
8
9    def __init__(self, k=3):
10     self.k = k
11
12   def fit(self, X, y):
13     self.X_train = X
14     self.y_train = y
15
16   def predict(self, X):
17     y_pred = [self._predict(x) for x in X]
18     return np.array(y_pred)
19
20   def _predict(self, x):
21     # 1. hitung jarak antara X ke semua data train
22     jarak_sample = [euclidean_distance(x, x_train) for x_train in self.X_train]
23     # 2. Urutkan dan Ambil K terdekat
24     k_idx = np.argsort(jarak_sample)[:self.k]
25     # 3. ambil label semua jarak terdekat
26     k_neighbor_labels = [self.y_train[i] for i in k_idx]
27     # 4. lakukan voting terbanyak
28     most_common = Counter(k_neighbor_labels).most_common(1)
29     return most_common[0][0]
30
```

Sebuah algoritma KNN membutuhkan sebuah hyperparameter K yang digunakan untuk proses prediksi. Pada proses prediksi, kita akan mengambil K jarak terdekat dari titik uji berdasarkan fungsi jarak. Selanjutnya baru dikembalikan hasil voting kelas terbanyak.

Pada pengujian digunakan dataset Iris (baris 8) dan evaluasi digunakan akurasi (baris 4-6). Pada baris ke 14, penulis menginisialisasi model dengan K=3. Lalu, baris ke 15 model dilatih berdasarkan data yang di dapat pada data splitting pada baris 11.

```
1  from sklearn import datasets
2  from sklearn.model_selection import train_test_split
3
4  def accuracy(y_true, y_pred):
5      accuracy = np.sum(y_true == y_pred) / len(y_true)
6      return accuracy
7
8  iris = datasets.load_iris()
9  X, y = iris.data, iris.target
10
11 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
12
13 k = 3
14 clf = KNN(k=k)
15 clf.fit(X_train, y_train)
16 predictions = clf.predict(X_test)
17 print("Akurasi KNN: ", accuracy(y_test, predictions))
```

```
Akurasi KNN:  0.9333333333333333
```

Pada barik ke 16 penulis melakukan prediksi terhadap datatest sehingga didapat kan akurasi sebesar 93.33%.

Jika anda ingin menggunakan metode penghitungan jarak yang lain misalnya haversine, maka anda cukup mengganti baris 4-5 dengan fungsi haversine maka anda telah menggunakan haversine sebagai metrik perhitungan jarak data.

4.4 Keunggulan dan Kelemahan KNN

KNN merupakan salah satu algoritma (model) pembelajaran mesin yang bersifat nonparametrik. Pembahasan mengenai **model parametrik** dan model **nonparametrik** bisa menjadi artikel sendiri, namun secara singkat, definisi model nonparametrik adalah model yang tidak mengasumsikan apa-apa mengenai distribusi instance di dalam dataset. Model nonparametrik biasanya lebih sulit diinterpretasikan, namun salah satu kelebihannya adalah garis keputusan kelas yang dihasilkan model tersebut bisa jadi sangat fleksibel dan nonlinear.

Selain itu, KNN mudah dipahami dan juga mudah diimplementasikan. Untuk mengklasifikasi *instance* x menggunakan KNN, kita cukup mendefinisikan fungsi untuk menghitung jarak antar-instance, menghitung jarak x dengan semua instance lainnya berdasarkan fungsi tersebut, dan menentukan kelas x sebagai kelas yang paling banyak muncul dalam k instance terdekat.

Adapun kekurangan-kekurangan dari KNN adalah

- Perlu menentukan hyperparameter K (jumlah tetangga terdekat) dan hyperparameter ini mempengaruhi kinerja KNN
- Tidak menangani nilai hilang (missing value) secara implisit. Jika terdapat nilai hilang pada satu atau lebih variabel dari suatu instance, perhitungan jarak instance tersebut dengan instance lainnya menjadi tidak dapat dilakukan. Bagaimana coba, menghitung jarak dalam ruang 3-dimensi jika salah satu dimensi hilang? Karenanya, sebelum menerapkan kNN kerap dilakukan imputasi untuk mengisi nilai-nilai hilang yang ada pada dataset. Contoh teknik imputasi yang paling umum adalah mengisi nilai hilang pada suatu variabel dengan nilai rata-rata variabel tersebut (*mean imputation*).
- Sensitif terhadap data pencilan (*outlier*). KNN bisa jadi sangat fleksibel jika k kecil. Fleksibilitas ini mengakibatkan kNN cenderung sensitif

terhadap data pencilan, khususnya pencilan yang terletak di "tengah-tengah" kelas yang berbeda.

- Rentan terhadap variabel yang non-informatif. Meskipun kita telah men-standard-kan rentang variabel, KNN tetap tidak dapat mengetahui variabel mana yang signifikan dalam klasifikasi dan mana yang tidak.
- Rentan terhadap dimensionalitas yang tinggi Berbagai permasalahan yang timbul dari tingginya dimensionalitas (data yang memiliki banyak variabel/*feature*) menimpa sebagian besar algoritma pembelajaran mesin, dan kNN adalah salah satu algoritma yang paling rentan terhadap tingginya dimensionalitas. Hal ini karena semakin banyak dimensi, ruang yang bisa ditempati instance semakin besar, sehingga semakin besar pula kemungkinan bahwa *nearest neighbour* dari suatu instance sebetulnya sama sekali tidak "near".

4.5 Studi Kasus – Klasifikasi Buah

Pada studi kasus ini, penulis akan mencoba melakukan klasifikasi buah berdasarkan ciri-cirinya. Dataset buah ini adalah dataset yang terdiri atas beberapa fitur yang dikumpulan oleh Mr. Iain Murray dari Universitas Edinburgh. Dataset ini berisi pengukuran fisik buah seperti apel, orange dan lemon.

Pada studi kasus ini, langkah pertama yang dilakukan adalah membaca file dataset. Adapun conoh isi dari dataset adalah sebagai berikut

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4
5  buah=pd.read_table('fruit_data_with_colors.txt')
6  buah.head()
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79

4.5.1 Memahami dataset

Pada dataset ini terdapat 7 kolom yaitu fruit label, fruit_name, fruit_subtype, mass, width, height dan color_score. Dataset ini terdiri atas 59 data buah yang terbagi atas tiga kelas yaitu apple, mandarin, orange dan lemon.

```
1   buah.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59 entries, 0 to 58
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   fruit_label    59 non-null     int64
 1   fruit_name     59 non-null     object
 2   fruit_subtype  59 non-null     object
 3   mass           59 non-null     int64
 4   width          59 non-null     float64
 5   height         59 non-null     float64
 6   color_score    59 non-null     float64
dtypes: float64(3), int64(2), object(2)
memory usage: 3.4+ KB
```

Pada dataset ini, terdapat kolom fruit label dan fruit name, dimana kedua kolom itu memiliki nilai sama dimana semua buah apple bernilai 1, mandarin bernilai 2, orange bernilai 3 dan orange bernilai 4.

```
1   #Kroscek hubungan label dan nama
2   label_mapping = dict(zip(buah.fruit_label.unique(), buah.fruit_name.unique()))
3   label_mapping
```

```
{1: 'apple', 2: 'mandarin', 3: 'orange', 4: 'lemon'}
```

4.5.2 Pembersihan Data dan Feature Enginerring

Langkah pertama pembersihan data adalah pengecekan missing data. Pada dataset ini tidak ditemukan ada data yang hilang.

```
1   #Checking for NaN values
2   buah.isna().sum()
```

```
fruit_label      0
fruit_name       0
fruit_subtype    0
mass             0
width            0
height           0
color_score      0
dtype: int64
```

Selanjutnya, karena fruit label dan fruit name adalah fitur yang sama maka kita drop beberapa kolom. Pada akhirnya kita hanya menggunakan 5 kolom

```
buah = buah.drop(["fruit_name","fruit_subtype"], axis=1)
buah.head()
```

	fruit_label	mass	width	height	color_score
0	1	192	8.4	7.3	0.55
1	1	180	8.0	6.8	0.59
2	1	176	7.4	7.2	0.60
3	2	86	6.2	4.7	0.80
4	2	84	6.0	4.6	0.79

Agar meningkatkan performa maka kita melakukan scaling ke semua fitur kecuali label

```
from sklearn import preprocessing
from sklearn import pipeline
from sklearn import model_selection
minmax_scaler = preprocessing.MinMaxScaler()
fruits_minmax_scaled = minmax_scaler.fit_transform(
    buah.drop(["fruit_label"], axis=1)
)
```

Setelah itu dataset tersebut dibagi menjadi data *training* dan data *test*. Pada kasus ini kita akan membuat 2 data training yaitu data training yang di scaling dan tidak. Hal ini bertujuan untuk membuktikan bagaimana pengaruh scaling terhadap performa, terutama untuk algoritma-algoritma berbasis *distance*

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

fruits_df_scaled = pd.DataFrame(fruits_minmax_scaled, columns=buah.columns[3:])
fruits_df_scaled.head()

X=buah[['mass','width','height']]
Y=buah['fruit_label']
X_train,X_test,y_train,y_test=train_test_split(X,Y,random_state=0)
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled = train_test_split(fruits_df_scaled,
                                                                                buah["fruit_label"],
                                                                                random_state=0)
```

4.5.3 Pelatihan dan evaluasi

Pertama-tama kita akan melakukan pelatihan pada dataset yang tidak di scaling. Kita akan menggunakan K=3 lalu melakukan training (baris 2) dan evaluasi (baris 3)

```
1  knn=KNeighborsClassifier(n_neighbors=3)
2  knn.fit(X_train,y_train)
3  knn.score(X_test,y_test)
```

```
0.5333333333333333
```

Tanpa scaling, model mendapatkan performa 53.33%, namun jika menggunakan data yang telah di *scaling*, performanya meningkat sebanyak 93.33%.

```
1  knn2=KNeighborsClassifier(n_neighbors=3)
2  knn2.fit(X_train_scaled,y_train_scaled)
3  knn2.score(X_test_scaled,y_test_scaled)
```

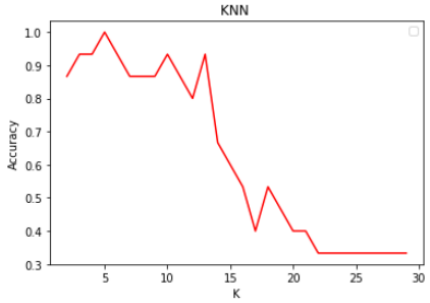
```
0.9333333333333333
```

Sebagai ilustrasi bandingkan misalkan data kita dengan fitur A yang bernilai 1sampai 10 dan fitur B bernilai 1 sampai 100. Titik x1 bernilai (1,10) dan x2 bernilai (2,20). Jika dilihat perbedaanya fitur A berbeda sebanyak 1 poin namun fitur B berbeda sebanyak 10 poin. Namun jika di scaling maka fitur A dan B memiliki perbedaan yang sama.

Langkah selanjutnya adalah pemilihan K terbaik. Sederhananya tinggal melakukan perulangan lalu dipetakan kedala grafis. Berdasarkan hasil percobaan, K terbaik adalah 5.ss

```
# mencari K Terbaik
K = np.arange(2,30)

scores = []

for k in K:
    model = KNeighborsClassifier(n_neighbors=k).fit(X_train_scaled,y_train_scaled)
    score = model.score(X_test_scaled,y_test_scaled)
    scores.append(score)

plt.plot(K,scores, c = "red")
plt.title("KNN ")
plt.ylabel("Accuracy")
plt.xlabel("K")
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



4.6 K-Dimensional Tree

K-Dimensional Tree (KD-Tree) merupakan salah satu solusi atas permasalahan performa yang dimiliki oleh KNN dimana, KNN akan membandingkan data test ke semua data latih. Jika data latih hanya 10 atau 100 tidak jadi masalah untuk melakukan ini, namun jika data berjumlah sangat banyak dan berdimensi tinggi maka hal ini akan menjadi permasalahan. Salah satu solusi yang digunakan adalah menggunakan KD Tree.

KD-Tree merupakan sebuah pohon biner (*Binary Tree*) yang telah di modifikasi sedemikian rupa sehingga dapat melakukan pencarian multidimensi pada data. Prinsip kerjanya adalah membuat partisi-partisi (sekat-sekat), lalu partisi tersebut digunakan untuk mengatur data-data sehingga mudah dilakukan pencarian. Singkatnya, dengan algoritma ini kita tidak perlu membandingkan dengan semua data, cukup dengan beberapa kandidat saja. Perbedaan KD-Tree dengan pohon biner adalah setiap node leaf adalah titik berdimensi K. Sebagai ilustrasi, data seorang karyawan dikatakan berdimensi 5 (K=5) maka data karyawan tersebut berisi 5 fitur yang tu no Karyawan, Umur, Gaji, Tahun Masuk, Jarak rumah.

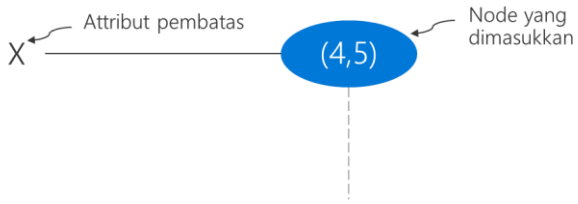
Pada Algortima ini, setiap node yang bukan leaf akan menghasilkan sebuah *hyperplane* yang akan memisahkan node-node yang berada pada subtree kanan. *Hyperplane* inilah yang berfungsi sebagai partisi node. Adapun langkah-langkah yang KD-Tree adalah sebagai berikut

1. Pilih attribut/fitur yang digunakan sebagai pembatas. Salah satu cara menggunakan axis (*axis = depth mod k*). Penggunaan mod pada axis bertujuan agar fitur dipilih secara bergantian, contoh data dengan 4 fitur maka pada depth tree 1 dipilih fitur 1, depth tree-2 dipilih fitur kedua, dan seterusnya. Cara lain seperti mencari fitur dengan range paling lebar juga menjadi opsi yang baik
2. Setelah itu data yang berada pada axis yang dipilih pada step 1 diurutkan, lalu hitung rata-rata atau media sebagai *pivot value*

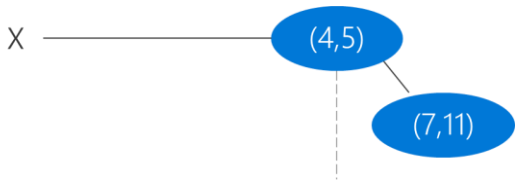
3. Buat node dan pecah data menjadi dua subtree sesuai aturan binary tree. Data yang nilainya kecil dari pivot value berada di partisi/sub-tree kiri dan nilai yang besar dari pivot value berada disebelah kanan. Ulangi step 1 sampai kedalaman yang ditentukan.

Sebagai contoh kita akan membuat KD-Tree dari data 2 dimensi (X,Y) berikut ini: (4, 5), (7, 11), (2, 7), (6, 12), (9, 1), (1, 4), (10, 19) . Adapun proses pembentukan K-dimensional tree adalah sebagai berikut:

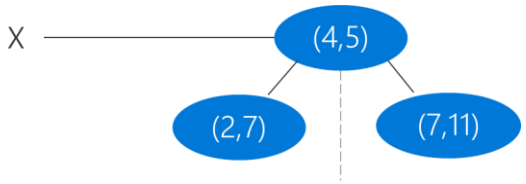
1. Pada langkah pertama maka insert langsung (4,5) karena tree masih kosong.



2. Insert (7,11). Karena nodenya sudah tidak kosong maka kita akan membandingkannya dengan (4,5) pada koordinat X. Karena 7 lebih besar dari 4 maka (7,11) diletakkan disebelah kanan

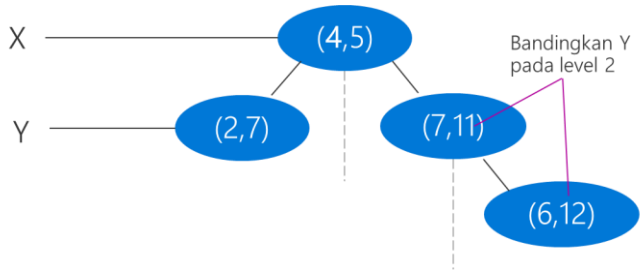


3. Insert (2,7). Langkah pertama cek dengan pembatas pada root (4,5). Karena 2 lebih kecil dari 4 maka langsung isi disebelah kiri (4,5).

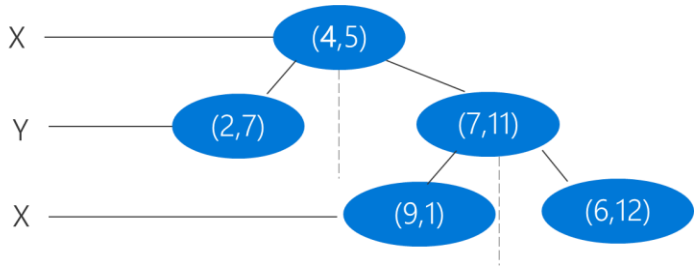


4. Insert (6, 12). Cek pembatas X pada (4,5), karena 6 lebeih besar dari 4 maka lanjutkan pengecekan di tree sebelah kanan pada pembatas Y (sebelumya di cek pada fitur X, sekarang diganti keY). Pada level 2 ini

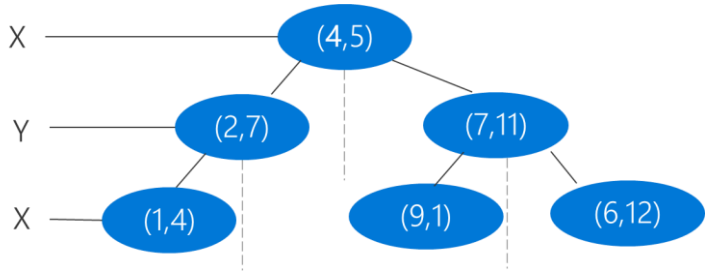
12 lebih besar dari 11 maka node akan diletakkan disebelah kanan (7,11)



5. Insert (9, 1). Sama seperti sebelumnya cek pembatas X pada level 1, 4 lebih besar dari 9, maka node berada disebelaj kanan (4,5). Pada level 2 sebelah kanan adalah (7,11) dengan pembatas y dimana 11 lebih besar dari 1 maka node berada disebelah kiri (7,11).

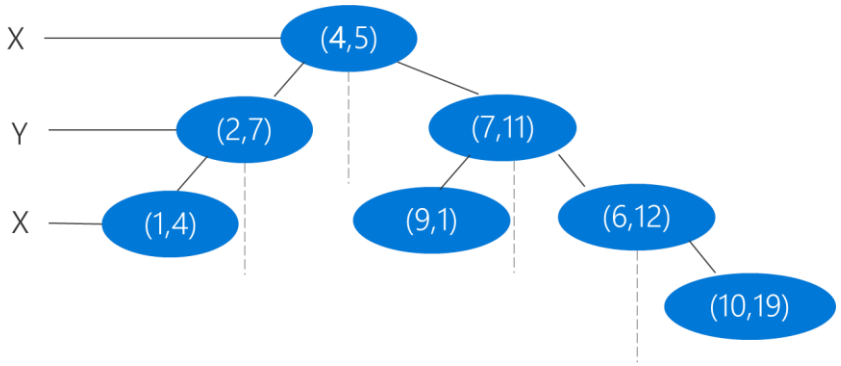


6. Insert (1, 4). Cek pembatas x pada (4,5), karena 1 lebih kecil maka, node berada disebelah kiri (4,5). Pada level 2, cek pembatas Y pada (2,7) karena (2,7) berada disebelah kiri (2,7). 4 lebih kecil dari 7 maka (1,4) berada disebelah kiri (2,7)

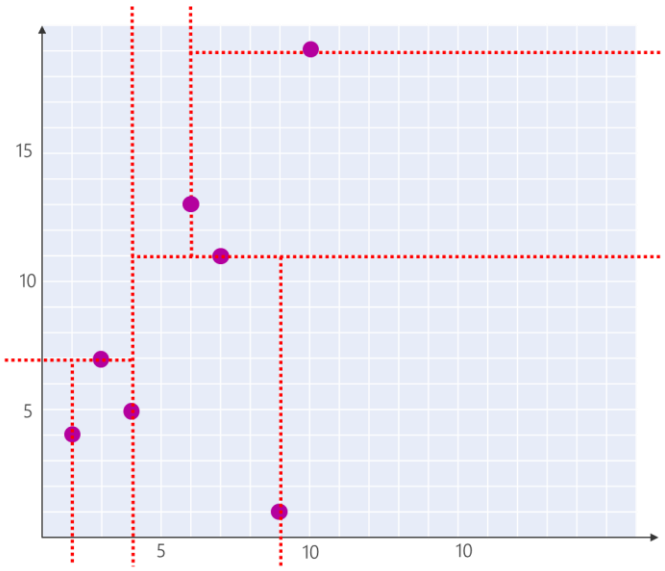


7. Insert (10, 19). Pada level 1, cek (4,5) pada pembatas x, karena 10 lebih besar dari 5 maka cek disebelah kanan (4,5). Pada level 2 cek pembatas y pada (7,11), karena 19 lebih besar maka node akan berada disebelah

kanan (7,11). Pada lebel 3 cek (6,12) pada pembatas X, dimana 10 lebih besar dari 6 maka (10,19) berada disebelah kanan.



Jika divisualisasikan, maka setiap langkah akan membagi area-area data sehingga proses pemilihan lebih cepat. Berikut ini visualisasi luaran partisi dari K-Dimensional Tree



Gambar 4.8. Ilustrasi *Haversine distance*

Bab

5

Regresi Linier

Tidak ada rasa bersalah yang dapat mengubah masa lalu dan tidak ada kekhawatiran yang dapat mengubah masa depan. – Umar bin Khattab.

Topik Pembelajaran:

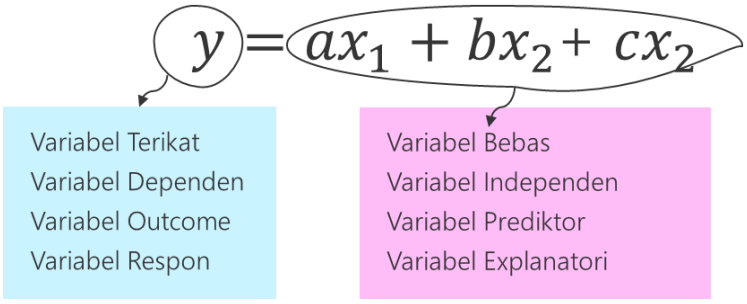
1. Definisi Regresi linear
2. Bagaimana langkah-langkah Implementasi implementasi regresi linear menggunakan Least Suare dan Gradient Descent
3. Regresi Linear berganda
4. Studi Kasus Regresi linear

Regresi adalah salah satu teknik dalam *Machine Learning*, dimana teknik ini berasal dari ilmu statistik untuk mencari parameter-parameter dalam persamaan linear yang dapat memetakan input dan output. Regresi linier merupakan teknik tertua, paling mudah dan banyak digunakan pada *supervised machine learning* untuk memprediksi sebuah nilai atau keadaan. Menurut penulis, teori ini merupakan salah satu teori paling penting abad ini. Tanpa teori ini, ilmu tentang Deep Learning tidak aka nada.

Analisis regresi setidak-tidaknya memiliki 3 kegunaan, yaitu mendeskripsi fenomena data atau kasus yang sedang diteliti; untuk tujuan kontrol; serta untuk tujuan prediksi. Regresi mampu mendeskripsikan fenomena data melalui terbentuknya suatu model hubungan yang

bersifatnya numerik. Inti dari regresi linier adalah memprediksi variable target dengan mencocokkan dengan fungsi relasi linier terbaik diantara variabel bebas dan terikat.

Untuk memahami konsep variabel bebas atau independen, seseorang harus memahami makna variabel. Variabel didefinisikan sebagai properti dari peristiwa atau objek tertentu. Variabel bebas adalah variabel yang dimanipulasi atau efek diukur dan dibandingkan. Nama lain untuk variabel bebas adalah variabel Prediktor atau variabel independen atau variabel explanatori. Variabel bebas digunakan memprediksi atau meramalkan nilai-nilai dari variabel terikat atau dependen dalam model.



Gambar 5.1. Ilustrasi nama-nama variable

Selain variabel bebas ada juga variabel terikat. Variabel ini nilainya sepenuhnya bergantung pada variabel bebas dengan kata lain, nilai variabel tersebut diprediksi atau diasumsikan oleh variabel bebas. Nama lain untuk variabel terikat adalah variabel outcome atau variabel dependen atau variable response.

Sebagai contoh, nilai seorang siswa dapat menjadi variabel terikat karena dapat berubah tergantung pada beberapa faktor, seperti seberapa banyak dia belajar, berapa banyak yang dia menghafal di malam hari sebelum dia melaksanakan tes, atau bahkan seberapa lapar dia ketika dia sedang ujian. Biasanya ketika seseorang mencari hubungan antara dua hal, seseorang mencoba mencari tahu apa yang membuat variabel terikat dapat berubah.

Dalam kasus model linear, kita memiliki persamaan umum sebagai: Y = aX + b, dimana Y adalah variabel terikat pada X, oleh karena itu, X, adalah variabel bebas.

5.1 Regresi Linier Sederhana

Seperti namanya, Regresi Linier Sederhana adalah sebuah metode sederhana untuk memprediksi nilai variabel dependen Y berdasarkan satu variabel dependen X. Secara matematis persamaan linier sederhana dapat dilihat pada persamaan 2.1

$$Y \approx b_0 + b_1 X \qquad (5\text{-}1)$$

Dimana:

- Y : atau *dependent variable*, adalah akibat dari suatu sebab, misalnya: kenaikan gaji karyawan (terhadap lama pengalaman kerja), atau tinggi/rendah ranking murid (terhadap lama waktu belajarnya).
- X : atau *independent variable*, adalah hal yang diasumsikan menjadi sebab atas suatu hal, yang nantinya bisa mengakibatkan Y.
- b_1 : atau *coefficient*, adalah suatu unit / proporsi yang dapat mengubah nilai "x"
- b_0 : atau *constant*, adalah nilai awal "x" pada suatu kejadian

Jika kita aplikasikan dalam contoh sehari-hari misalnya total penjualan yang dihasilkan oleh seorang sales berbanding lurus dengan banyaknya dia melakukan iklan maka dapat dirumuskan

- Y sebagai hasil penjualan dimana ada faktor lain yang mempengaruhi hasil penjualan tersebut
- X sebagai iklan yang dilakukan untuk mempengaruhi hasil penjualan

$$\boldsymbol{Nilai\ Penjualan} \approx b_0 + b_1 * \boldsymbol{Iklan}$$

Pada persamaan tersebut nilai $b_0$dan b_1 belum diketahui dan nilai tersebut akan bersifat sebagai konstanta (nilainya tetap). Ketika dilakukan

proses training (mempelajadi data yang ada) maka kita dapat menentukan nilai $b_0$dan b_1. Apabila nilai $b_0$dan b_1 telah diketahui, maka kita dapat menggunaan model tersebut untuk melakukan prediksi

5.1.1 Solusi Regresi Linear Menggunakan *Least Square*

Untuk dapat melakukan prediksi, kita perlu mengetahui nilai $b_0$dan b_1. Oleh karena itu kita dapat menemukan nilai-nilai tersebut dengan melakukan observasi terhadap data-data yang ada. Nilai-nilai koefisien tersebut dapat diperoleh dengan menggunakan metode *Least Square***.** Prinsip dasar dari *least square* adalah menemukan b yang memiliki error kuadrat yang paling kecil (minimum). Misalnya kita memiliki sejumlah n data yang merepresentasikan n pasang data dimana setiap pasangnya berisi variabel X (Iklan) dan Y (Nilai Penjualan)

$$(x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots\ldots (x_n, y_n)$$

Nilai-nilai $b_0$dan $b_1$dapat dihitung dengan menyubstitusi nilai x dan y serta mencari rata-rata x dan y ke dalam persamaan 2 dan 3.

$$b_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \quad (5\text{-}2)$$

$$b_0 = \bar{y} - b_1 \bar{x} \quad (5\text{-}3)$$

Dimana

x_i: nilai ke-i variabel x

y_i: nilai ke-i variabel y

$\bar{x}$: rata-rata dari variabel x

$\bar{y}$: rata-rata dari variabel y

5.1.2 Contoh soal dan Implementasi metode *Least Square*

Soal:

Seorang penjual ingin mempelajari hubungan antara promosi dan jumlah penjualan. Penjual tersebut kemudian mengambil data selama 10 bulan terakhir terhadap yang menunjukkan nilai antara promosi dan jumlah penjualan. Adapun data penjualan dan jumlah iklan adalah sebagai berikut.

Tabel 1. Data Promosi dan Penjualan

Jumlah Promosi	Total Penjualan
1	270
2	350
3	500
4	700
5	800
6	850
7	900
8	900
9	1000
10	1200

Dengan menggunakan data tabel 3, tentukan model menggunakan regresi linear dan prediksi jumlah penjualan jika jumlah promosi yang dilakukan adalah 0, 11, 12?

Penyelesaian :

Penyelesaiannya mengikuti Langkah-langkah dalam sebagai berikut:

1. Tentukan Variabel Dependen dan Independen

   Variabel Independen (X) : Jumlah Promosi,

   Variabel Dependen (Y) : Total Penjualan

2. Hitung Koefisien regresi linier

Berdasarkan data tabel 2 didapatkan rata-rata jumlah promosi ($\bar{x}$) sebesar 5.5 dan rata-rata total penjualan ($\bar{y}$) sebesar 747. Untuk mempermudah penghitungan maka dapat menggunakan tabel 3 untuk membantu penghitungan.

Tabel 2. Data perhitungan koefisien regresi linier

X	Y	$x_i - \overline{x}$	$y_i - \overline{y}$	$(x_i - \overline{x})(y_i - \overline{y})$	$(x_i - \overline{x})^2$
1	270	-4,5	-477	2146,5	20,25
2	350	-3,5	-397	1389,5	12,25
3	500	-2,5	-247	617,5	6,25
4	700	-1,5	-47	70,5	2,25
5	800	-0,5	53	-26,5	0,25
6	850	0,5	103	51,5	0,25
7	900	1,5	153	229,5	2,25
8	900	2,5	153	382,5	6,25
9	1000	3,5	253	885,5	12,25
10	1200	4,5	453	2038,5	20,25
			TOTAL	7785	82,5

Berdasarkan tabel maka gunakan persamaan 5-2 dan 5-3 untuk menentukan nilai koefisien-koefisien tersebut

$$b_1 = \frac{7785}{82,5} = 94.36$$

$$b_0 = 742 - 94.36 * 5.5 = 228$$

3. Buat model persamaan regresi

   Dari nilai-nilai koefisien didapatkan model sebagai berikut

   Y = 228.0 + 94.36 * X

4. Lakukan prediksi

   Untuk melakukan prediksi maka kita harus menyubstitusi nilai X.

   Prediksi jika jumlah promosi = 0 adalah Y = 228.0 + 94.36 * 0 = 228
   Prediksi jika jumlah promosi = 11 adalah Y = 228.0 + 94.36 * 11 = 1265,96
   Prediksi jika jumlah promosi = 12 adalah Y = 228.0 + 94.36 * 12 = 1360,32

Implementasi Python:.

Untuk menentukan koefisien kita dapat menggunakan library Numpy dan Matplotlib untuk menghitung nilai koefisien berdasarkan persamaan 2 dan 3.

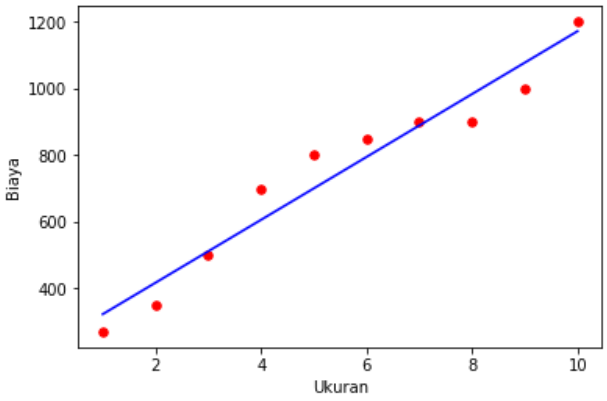
Source Code 2. Implementasi Regresi Linier Sederhana menggunakan Least Square.

```
1.  import matplotlib.pyplot as plt
2.  import numpy as np
3.
4.  def estimasi_koofisien(x, y):
5.      # mendapatkan jumlah data atau item yang observasi
6.      n = np.size(x)
7.      # rata-rata x dan y
8.      mean_x= np.mean(x)
9.      mean_y= np.mean(y)
10.     # hitung cross-deviation dan deviation x
11.     numerator = 0
12.     denominator = 0
13.     for i in range(n):
14.         numerator += (x[i] - mean_x) * (y[i] - mean_y)
15.         denominator += (x[i] - mean_x) ** 2
16.     b1 = numerator/ denominator
17.     b0 = mean_y - (b1 * mean_x)
18.
19.     return(b0, b1)
20.
21. def grafik_hasil (x, y, b):
22.     # plot titik yang ada di dataset
23.     plt.scatter(x, y, color = "r",marker = "o", s = 30)
24.     # buat persamaan model
25.     y_pred = b[0] + b[1]*x
26.     # plot persamaan model
27.     plt.plot(x, y_pred, color = "b")
28.     # label untuk x dan Y
29.     plt.xlabel('Ukuran')
30.     plt.ylabel('Biaya')
31.     # fungsi untuk menampilkan graph
32.     plt.show()
33.
34. def main():
35.     # dataset dari tabel 3
36.     x = np.array([1,2,3,4,5,6,7,8,9,10])
37.     y = np.array([270,350,500,700,800,850,900,900, 1000,1200])
38.
39.     b = estimasi_koofisien(x, y)
40.     print("Koofisien:\nb0 = {} \nb1 = {}".format(b[0], b[1]))
41.     print("Fungsi regresi Y = {}X + {}".format(b[0], b[1]))
42.     # plotting regression line
43.     Grafik_hasil(x, y, b)
```

```
44.
45. if __name__ == "__main__":
46.     main()
47.
48.
```

Keluaran

```
b0 = 228.0
b1 = 94.36363636363636
Fungsi regresi Y = 228.0X + 94.36363636363636
```



Baris ke 4 sampai 21 merupakan implementasi dari persamaan 2 dan 3. Fungsi estimasi koefisien ini membutuhkan dua parameter yaitu X dan Y. X merupakan variabel bebas dan Y merupakan variabel terikat. Variabel tersebut harus dalam format array.

Pertama-tama (baris ke 6) dihitung berapa jumlah dataset yang dimasukkan melalui variabel X. Lalu dicari rata-rata X dan Y (baris ke 8 dan 9). Jika kita perhatikan persamaan 2, maka di sana ada lambang Σ (sigma) yang berarti kita akan melakukan perulangan sebanyak jumlah data (baris 13), lalu melakukan penjumlahan $(x_i - \bar{x})(y_i - \bar{y})$ disimpan dalam variabel numerator (baris 14) dan penjumlahan $(x_i - \bar{x})^2$ disimpan dalam variabel denumerator (baris 15). Lalu setelah semua data dihitung koefisien b_1 diperoleh dengan membagi numerator dan denumerator (baris 16). setelah mendapatkan nilai b_1 maka penghitungan dilanjutkan untuk persamaan 3 sehingga diperoleh variabel b_0 (baris 17).

Untuk melihat hasil akhir, dilakukan pemetaan titik dan model yang dihasilkan sehingga terlihatlah relasi antara model dan titik.

5.1.3 Regresi Linear Menggunakan *Gradient Descent.*

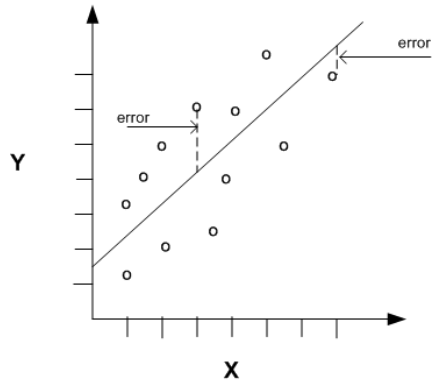
Algoritma *gradient descent* atau penurunan gradien merupakan algoritma yang digunakan untuk mencari nilai minimum lokal yang dapat dihasilkan dari suatu fungsi parametrik. Teknik ini didasarkan pada fakta bahwa nilai gradien dari suatu fungsi pada titik tertentu menyatakan kemiringan lereng dari nilai tersebut terhadap titik di sekitarnya sehingga nilai minimum dapat diraih dengan mengurangi nilai titik tersebut dengan nilai gradien. Metode ini biasanya digunakan ketika parameter sistem tidak dapat dikalkulasikan dan harus dilakukan pencarian dengan metode optimasi.

Sederhananya, *gradient descent* adalah algoritma yang meminimalkan sebuah fungsi. Fungsi tersebut akan diberikan sejumlah parameter awal, lalu algoritma ini akan memulai dengan menghitung *error* dari nilai awal tersebut dan secara iteratif mengubah nilai awal parameter tadi sehingga *output* dari fungsi tersebut menjadi minimal.

Pada kasus regresi linear, fungsi yang terbaik adalah fungsi yang memiliki *error* terkecil. *Error* dihitung dari jarak antara titik ke model (berdasarkan nilai parameter). Tujuan dari algoritma *gradient descent* menemukan parameter yang menghasilkan error terkecil pada semua titik. Dengan kata lain metode ini mencari parameter mana yang memiliki error terkecil menggunakan iterasi.

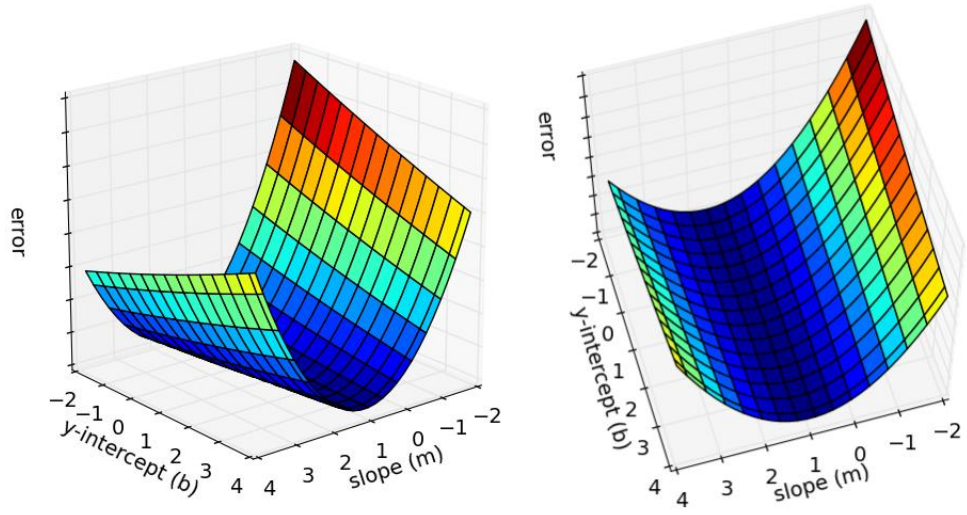
Dengan asumsi bahwa fungsi model mengikuti persamaan `y = mx + b`, dimana m adalah kemiringan atau gradien dan b adalah y-intercept, maka untuk menghitung error, harus dilakukan iterasi pada seluruh titik `(x,y)` yang ada pada dataset lalu dicari rata-rata jaraknya. Adapun formula untuk mencar error pada sebuah model dengan parameter m dan b seperti pada gambar 17 dapat dilihat pada persamaan 4.

$$Error_{(m,b)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2 \quad (5\text{-}4)$$



Gambar 5.2. Ilustrasi error pada model regresi linear

Model garis terbaik akan memberikan rata-rata nilai error terkecil. Jika kita visualiasikan secara 2D antara parameter m dan b dengan error maka akan didapat visualisasi seperti gambar 18.



Gambar 5.3. Mapping parameter model dan error

Setiap titik pada Gambar 5.3 merepresentasikan garis atau model. Ketika pencarian nilai error terendah, kita akan mulai pada satu titik lalu bergerak ketitik lainnya yang memiliki error yang lebih kecil (bergerak turun). Kemiringan adalah petunjuk kita dalam mencari error terendah. Untuk menghitungnya kita dapat menggunakan turunan dari fungsi error. Karena

fungsi error memiliki dua parameter maka prosesnya dibagi sesuai dengan parameter. Adapun turunan dari fungsi error adalah sebagai berikut

$$\frac{\partial}{\partial m} = \frac{2}{N}\sum_{i=1}^{N} -x_i(y_i - (mx_i + b)) \quad (5\text{-}5)$$

$$\frac{\partial}{\partial b} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (mx_i + b)) \quad (5\text{-}6)$$

Setelah mengetahui bagaimana cara untuk melakukan pencarian error terkecil maka, algoritma ini membutuhkan dua parameter lagi yaitu epoch dan learning rate. Sederhananya epoch adalah jumlah iterasi dan *learning rate* adalah nilai yang diperlukan untuk berpindah dari satu titik ke titik lainnya.

5.1.4 Implementasi Regresi Linier menggunakan Gradient Descent

Fungsi estimasi_gs pada baris ke-4 adalah fungsi yang mengimplentasikan gradient descent. Pada fungsi tersebut dibutuhkan 6 parameter yaitu X (nilai variabel independen), y (nilai variabel dependen), b0 dan b1 (nilai awal parameter), epoch dan learning rate. Hasil dari algortima ini dibandingkan algortima OLS (dengan data yang sama) memiliki perbedaan tetapi tidak signifikan.

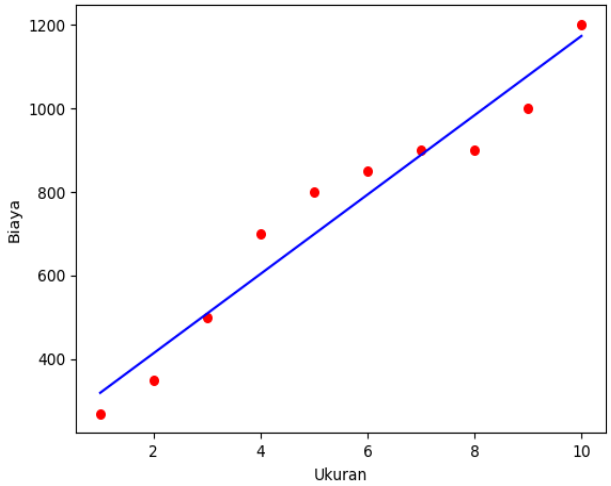
Source Code 3. Implementasi Regresi Linier Sederhana menggunakan Gradient Descent

```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.
4.  def estimasi_gs(X,y,b0=0,b1=0,epochs=100,learning_rate=0.01 ):
5.       N = float(len(y))
6.       for i in range(epochs):
7.          y_current = (b0 * X) + b1
8.          cost = sum([data**2 for data in (y-y_current)]) / N
9.          b1_gradient = -(2/N) * sum(X * (y - y_current))
10.         b0_gradient = -(2/N) * sum(y - y_current)
11.         b0 = b0 - (learning_rate * b1_gradient)
12.         b1 = b1 - (learning_rate * b0_gradient)
13.      return b1, b0, cost
14.
15. def grafik_hasil(x, y, b):
16.     # plotting nilai titik
17.     plt.scatter(x, y, color = "r",marker = "o", s = 30)
18.     # protting nilai hasil prediksi
19.     y_pred = b[0] + b[1]*x
20.     # plotting garis regresi
21.     plt.plot(x, y_pred, color = "b")
22.     # label untuk x dan Y
23.     plt.xlabel('Ukuran')
24.     plt.ylabel('Biaya')
25.
26.     # fungsi untuk menampilkan graph
27.     plt.show()
28. def main():
29.     # Datasets yang akan digunakan
30.     x = np.array([ 1,2,3,4,5,6,7, 8, 9,10])
31.     y = np.array([270,350,500,700,800,850,900,900,1000,1200])
32.     g = estimasi_gs(x,y,0,0,10000,0.001)
33.
34.     print("Koofisien:\nb0 = {} \nb1 = {}".format(g[0], g[1]))
35.     print("Fungsi regresi Y = {}X + {}".format(g[0], g[1]))
36.
37.     # plotting regression line
38.     grafik_hasil(x, y, g)
```

```
39.
40. if __name__ == "__main__":
41.     main()
42.
```

Keluaran

```
b0 = 227.99999999
b1 = 94.36363636363636
Fungsi regresi Y = 228.0X + 94.36363636363636
```



5.2 Regresi Linier Berganda

Regresi linear berganda mirip dengan regresi linear sederhana, perbedaannya adalah Regresi linear berganda memiliki lebih dari satu variabel bebas untuk memprediksi sebuah variabel terikat. Adapun bentuk persamaannya adalah

$$Y \approx b_0 + b_1X_1 + b_2X_2 + \cdots + b_nX_n \qquad (5\text{-}7)$$

Dimana:

- Y : variabel terikat
- X_1, X_2, X_3, X_n : variabel bebas.
- b_1, b_2, b_3, b_n : koefisien variabel bebas X
- b_0 : atau *constant*, adalah nilai awal "x" pada suatu kejadian

Bertambahnya jumlah variabel X hingga lebih dari satu sangat memungkinkan, karena dalam kasus pada kehidupan nyata, semua faktor-faktor atau variabel-variabel saling berkaitan satu dengan lainnya. Sebagai contoh dalam ilmu ekonomi, inflasi tidak hanya dipengaruhi oleh variabel bunga deposito saja, tetapi sangat mungkin dipengaruhi oleh variabel lainnya seperti perubahan nilai tukar (kurs), jumlah uang beredar, kelangkaan barang, dan lain-lain.

5.3 Studi Kasus Regresi Linear Berganda

P.T. XYZ melakukan promosi sejumlah barang dagangan di outlet-outlet yang berbeda di seluruh Indonesia. Tujuannya selain untuk meningkatkan pendapatan, juga mengetahui pengaruh biaya promosi, jumlah *outlet*, laju pertumbuhan penduduk, jumlah pesaing, dan *income* terhadap penjualan. Tabel 5 menunjukkan data penjualan, biaya promosi, jumlah *outlet*, laju pertumbuhan penduduk, jumlah pesaing, dan *income* masyarakat di 15 cabang di Indonesia.

Tabel 3. Data hasil penelitian PT. Cemerlang

Daerah	Penjualan	Promosi	Outlet	Laju Penduduk	Pesaing	Income
Jakarta	205	26	159	2,00	15	5,46
Tangerang	206	28	164	1,50	16	2,43
Bekasi	254	35	198	1,75	19	2,56
Bogor	246	31	184	1,64	17	3,55
Bandung	201	21	150	2,65	11	4,35
Semarang	291	49	208	1,45	24	3,65
Solo	234	30	184	1,67	16	3,44
Jogya	209	30	154	2,74	10	2,55
Surabaya	204	24	149	1,35	14	4,79
Purwokerto	216	31	175	2,13	14	2,53
Madiun	245	32	192	2,64	11	2,75
Tuban	286	47	201	1,63	19	2,53
Malang	312	54	248	2,53	21	3,51
Kudus	265	40	166	2,54	18	2,81
Pekalongan	322	42	287	1,53	18	3,01

Sumber data:https://rochsunmkes.wordpress.com/

Berdasarkan tabel maka dapat dibentuk persamaan model yang akan digunakan untuk memprediksi nilai penjualan adalah sebagai berikut

$$Penjualan = b_0 + b_1(promosi) + b_2(Outlet) + b_3\left(Laju_{pend}\right) + b_4(Pesaing) + b_5(Income)$$

Untuk menyelesaikan permasalahan ini kita akan menggunakan library SciKit untuk melakukan pelatihan dan evaluasi model.

Source Code 4. Implementasi Regresi Linier Sederhana menggunakan Gradient Descent

```
1.  import numpy as np
2.  import pandas as pd
3.  import matplotlib.pyplot as plt
4.  from sklearn.model_selection import train_test_split
5.  from sklearn.linear_model import LinearRegression
6.  from sklearn import metrics
7.
8.  dataset = pd.read_csv('dataset_penjualan.csv')
9.  #dataset.head()
10. print(dataset.describe() )
11.
12. #persiapan data
13. X = dataset[['Penjualan','PromosiOutlet','LajuPenduduk','Pesaing']]
14. y = dataset['Income']
15.
16. X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=0)
17.
18. #Training the Algorithm
19. regressor = LinearRegression()
20. regressor.fit(X_train, y_train)
21. coeff_df = pd.DataFrame(regressor.coef_, X.columns,
22. columns=['Coefficient'])
23. print(coeff_df)
24.
25. y_pred = regressor.predict(X_test)
26.
27. df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
28. print(df)
29.
30. #Evaluasi model
31. print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,
    y_pred))
32. print('Mean Squared Error:', metrics.mean_squared_error(y_test,
    y_pred))
33. print('Root Mean Squared Error:',
    np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
34.
```

Keluaran

	Penjualan	PromosiOutlet	LajuPenduduk	Pesaing	Income
count	15.000000	15.000000	15.000000	15.00000	15.000000
mean	34.666667	187.933333	1.983333	16.20000	3.328000
std	9.678154	38.087243	0.507003	3.87667	0.922134

min	21.000000	149.000000	1.350000	10.00000	2.430000
25%	29.000000	161.500000	1.580000	14.00000	2.555000
50%	31.000000	184.000000	1.750000	16.00000	3.010000
75%	41.000000	199.500000	2.535000	18.50000	3.600000
max	54.000000	287.000000	2.740000	24.00000	5.460000

Coefficient

Penjualan	-0.125808
PromosiOutlet	0.005263
LajuPenduduk	0.866334
Pesaing	0.273363

Actual	Predicted	
206	2.43	3.549892
234	3.44	3.550823
204	4.79	3.297494

Mean Absolute Error : 0.9077401314408511

Mean Squared Error : 1.1646710300229983

Root Mean Squared Error: 1.0791992540874917

5.4 Studi Kasus Prediksi Penggunaan Bahan Bakar Mobil

Dataset Auto-Mpg merupakan data yang berkaitan dengan konsumsi bahan bakar pada kendaraan bermotor adapun variable (mil per galon). Dataset ini terdiri atas 9 kolom yaitu MPG (miles per gallon), Cylindes , Displacement, Horsepower, weight, acceleration, model year, origin dan car name. Dengan dataset ini kita ditantang untuk menenentukan berapa jumlah konsumsi bahan bakar sebuah kendaraan (miles per gallon). Adapun contoh isi dataset ini adalah

ID	mpg	cylinders	Displacement	Horsepower	weight	Acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

Adapun langkah-langkah yang dilakukan adalah

1. **Data Preprocessing (Persiapan Data)**

Pada data processing penulis akan melakukan pembersihan terhadap data dan feature extraction. Adapun langkah-langkah tersebut adalah

- Langkah pertama load library yang dibutuhkan yaitu, pandas, numpy, seaborn dan matplotlib.

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

- Setelah itu, langkah selanjutnya adalah membaca/loading dataset lalu menampilkan Informasi masing-masing fitur data dengan menggunakan perintah read_csv dan fungsi info()

```
[2]  1  # Baca Dataset MPG
     2  df = pd.read_csv('auto-mpg.csv')
     3  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64
 7   origin        398 non-null    int64
 8   car name      398 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

Berdasarkan informasi awal ini, ada hal yang aneh yaitu ***horsepower* bertipe object**. Logikanya *horsepower* harus bertipe float64 atau int64 karena pasti bernilai angka, namun memiliki tipe yang sama dengan car name.

- Langkah selanjutnya adalah mengivestigasi nilai unik horsepower. Dengan menggunakan fungsi unique, penulis menampilkan semua isi dari horsepower untuk mengetahui apa yang terjadi.

```
[3]  1  df['horsepower'].unique()
```

```
array(['130', '165', '150', '140', '198', '220', '215', '225', '190',
       '170', '160', '95', '97', '85', '88', '46', '87', '90', '113',
       '200', '210', '193', '?', '100', '105', '175', '153', '180', '110',
       '72', '86', '70', '76', '65', '69', '60', '80', '54', '208', '155',
       '112', '92', '145', '137', '158', '167', '94', '107', '230', '49',
       '75', '91', '122', '67', '83', '78', '52', '61', '93', '148',
       '129', '96', '71', '98', '115', '53', '81', '79', '120', '152',
       '102', '108', '68', '58', '149', '89', '63', '48', '66', '139',
       '103', '125', '133', '138', '135', '142', '77', '62', '132', '84',
       '64', '74', '116', '82'], dtype=object)
```

Ternyata ada horsepower yang bernilai "?". Hal ini dapat diartikan sebagai MISSING VALUE. Berikut ini adalah data-data yang bernilai "?"

```
[4]  1  df[df['horsepower'] == '?']
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

- Karena jumlah missing value sedikit, maka penulis akan mengabaikan data ini. Jadi record yang berisi data kosong ini dihapus saja, sehinga data yang awalnya 398 berkurang menjadi 392.

```
[10]  1  #menghilangkan semua MISSING value dan karakter "?"
      2  df = df.dropna()
      3  df = df[df['horsepower'] != '?']
      4  # memperbaiki tipe data horse power
      5  df['horsepower'] = pd.to_numeric(df['horsepower'])
      6  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    int64  
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
 6   model year    392 non-null    int64  
 7   origin        392 non-null    int64  
 8   car name      392 non-null    object 
dtypes: float64(3), int64(5), object(1)
memory usage: 30.6+ KB
```

- Selanjutnya, penulis akan melakukan Seleksi Fitur. Oleh karena itu akan dilihat apa saja isi dari dataset dan bagaimana korelasinya. Penulis menggunakan pairplot

```
[6]  1  sns.pairplot(df)
     2  plt.show()
```

Berdasarkan grafik tersebut, kolom model year, dan origin sangat jelas tidak berpengaruh terhadap kolom MPG jadi kolom tersebut dapa langsung dihilangkan (tidak digunakan). Selanjutnya pada kolom acceleration, korelasinya dengan kolom MPG tidak terlalu kuat, jadi penulis tidak menggunakan kolom tersebut.

Oleh karena ini dipilihlah 4 Fitur yang akan digunakan yaitu cylinders, displacement, horsepower dan weight

```
[13]  1  #pemilihan fitur 'cylinders', 'displacement', 'horsepower', 'weight'
      2  df_features = df[['cylinders', 'displacement', 'horsepower', 'weight']]
      3  df_features = df_features.apply(pd.to_numeric)
      4
      5  df_labels = df['mpg']
```

- Karena kita di Indonesia lebih familiar dengan Kilogram dibandingkan pound (LBS) maka penulis melakukan konversi terhadap data menjadi kilogram dan miles pergalon menjadi KM perliter. 1 LBS setara dengan 0.4535923 Kg, Miles setara dengan 1.60934 KG dan 1 gallon setara dengan 3.78541 Liter. Untuk konversi LBS ke liter tinggal mengalikan dengan 0.4535923 dan Miles per Gallon ke KM/L mengalikan dengan 1.60934/3.78541

```
[14]  1  #transformasi data
      2  #di indonesia kita mau pakai KG
      3  df_features['weight'] =df_features['weight']  * 0.4535923
      4  #Transformasi M/G (Miles per Galon) menjadi KM/L
      5  df_labels=df_labels * 0.4251437
```

- Selanjutnya adalah normalisasi data agar ukuran data seragam, perlu diingat bahwa metode ini sangat sensitive dengan outlier jadi normalisasi adalah salah satu cara untuk mengurangi resiko tersebut.

```
[16]  1  # normalize features
      2  df_features = (df_features - df_features.mean()) / df_features.std()
      3
```

- Langkah terakhir pada fase ini adalah pembagian data menjadi data latih dan data uji dengan porsi 80% dan 20%

```
[17]  1  from sklearn.model_selection import train_test_split
      2  X_train, X_test, y_train, y_test = train_test_split(
      3          df_features.values,
      4          df_labels.values,
      5          test_size=0.2)
```

2. Pelatihan Model

   Model akan dilatih menggunakan Algoritma Regresi Linear dari Scikit-Learn.

```
[18]  1  from sklearn.linear_model import LinearRegression
      2
      3  model = LinearRegression()
      4  model.fit(X_train, y_train)
      5
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

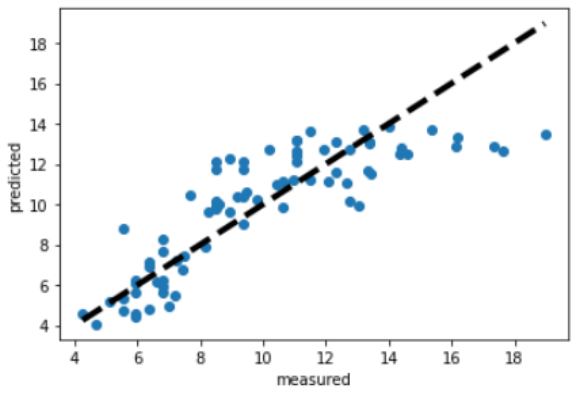
3. Evaluasi Model

   Model akan dievaluasi menggunakan mean square error dan R square.

```
[19]  1  from sklearn.metrics import mean_squared_error, r2_score
      2  y_predicted = model.predict(X_test)
      3
      4  print("Mean squared error: %.2f" % mean_squared_error(y_test, y_predicted))
      5  print('R²: %.2f' % r2_score(y_test, y_predicted))
      6
```

```
Mean squared error: 3.37
R²: 0.70
```

```
[20]  1  #plot perbandingan prediksi dan nilai asli
      2  fig, ax = plt.subplots()
      3  ax.scatter(y_test, y_predicted)
      4  ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
      5  ax.set_xlabel('measured')
      6  ax.set_ylabel('predicted')
      7  plt.show()
```



Bab

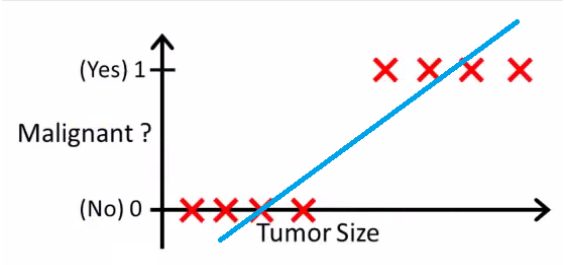
6

Regresi Logistik

Orang yang suka berkata jujur mendapatkan tiga hal: kepercayaan, cinta dan rasa hormat. – Ali bin Abi Thalib.

Regresi Logistik adalah metode yang bersumber dari ilmu statistik yang diadopsi oleh *Machine Learning* untuk melakukan klasifikasi. Metode ini biasanya digunakan untuk menyelesaikan kasus-kasus klasifikasi biner dimana masalah-masalah yang dihadapi hanya memiliki dua nilai kelas. Metode ini menggunakan beberapa variabel prediktor, baik numerik maupun kategori. Misalnya, probabilitas bahwa orang yang menderita serangan jantung pada waktu tertentu dapat diprediksi dari informasi usia, jenis kelamin, dan indeks massa tubuh. Regresi logistik juga digunakan secara luas pada bidang kedokteran dan ilmu sosial, maupun pemasaran seperti prediksi kecenderungan pelanggan untuk membeli suatu produk atau berhenti berlangganan.

Sebagai contoh, kita ingin memprediksi tumor ganas berdasarkan ukurannya. Semakin besar ukurannya maka tumor dapat dikatakan tumor ganas. Jika digambarkan dalam bentuk grafis, dengan mengganti nilai ganas dan tidak menjadi 1 dan 0 maka didapatkan seperti pada gambar 19. Jika permasalahan ini dipecahkan menggunakan metode linier maka model yang dihasilkan tidak akurat karena variabel dependennya bernilai hanya 0 atau 1. Fungsi linier tidak cocok untuk mengatasi permasalahan ini, salah satu fungsi yang cocok adalah fungsi logistik



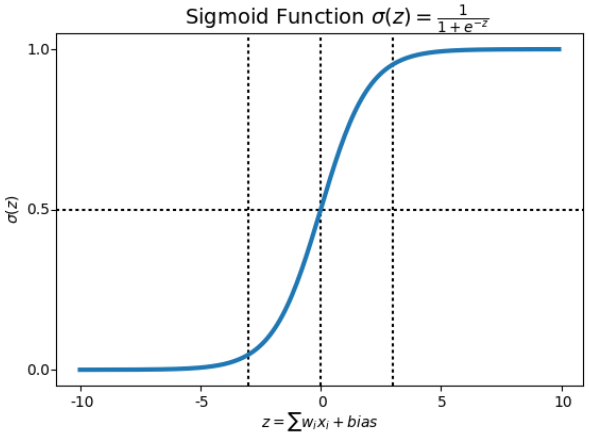
Gambar 6.1. Ilustrasi penggunaan regresi untuk prediksi

6.1 Fungsi Logistik

Regresi logistik diambil namanya dari fungsi yang digunakan yaitu fungsi logistik. Fungsi logistik ini sering juga disebut fungsi sigmoid yang dikembangkan oleh ahli statistika untuk mendeskripsikan pertumbuhan populasi pada kasus ekologi. Kurva yang dihasilkan berbentuk S yang bernilai di antara 0 dan 1. Adapun formula untuk sigmoid adalah

$$Y = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}} \qquad (6\text{-}1)$$

Dari fungsi 11 jika di plot nilai x dari -8 sampai dengan 8 maka akan ditransformasikan menjadi angka diantara 0 dan 1 seperti pada Gambar 6.2.



Gambar 6.2. Grafik Fungsi Sigmoid

6.2 Persamaan Regresi Logistik

Secara teknis, regresi logistik merupakan regresi linear yang substitusi ke dalam fungsi logistik, maka representasi persamaan regresi logistik dapat dilihat pada persamaan 6.1.

$$Y = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n)}} \quad (6\text{-}2)$$

Dimana Y adalah *output* yang akan diprediksi, sedangkan $b_0$sampai $b_n$adalah koefisien, dan x adalah variabel-variape independennya. Beberapa keuntungan dari penggunaan regresi logistic diantaranya:

- Mudah digunakan. Proses pelatihan dan penggunaan model cukup mudah dan tidak membutuhkan banyak maintenance.
- Interpretability. Modelnya mudah untuk diinterpretasikan.
- Scalability. Model algoritma ini dapat dilatih dengan cepat dan efisien serta membutuhkan daya komputasi yang rendah

6.3 Implementasi

Implementasi regresi logistic menggunakan Scikit-Learn sangat sederhana. Pada library tersebut, juga dimplementasikan regresi logistic dengan prinsip OVR (One versus Rest) dan Maximum Entropi.

Langkah pertama adalah memanggil library yang akan digunakan yaitu LogisticRegression dari sklearn.linear_model

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  from sklearn.linear_model import LogisticRegression
4  from sklearn.metrics import classification_report, confusion_matrix
```

Setelah itu kita akan membuat 10 data dan 10 label sebagai data latih sekaligus data uji.

```
1  x = np.arange(10).reshape(-1, 1)
2  y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

Selanjutnya adalah membuat model model regresi dan melatihnya

```
#membuat model
model = LogisticRegression( random_state=0)
#Pelatihan Model
model.fit(x, y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Hasil pelatihan dapat dilihat melalui attribute classes_, intercept_ dan coef_

```
print("Kelas :", model.classes_)
print("Koefisien Model :", model.coef_)
print("Intercept Model :", model.intercept_)
```

```
Kelas : [0 1]
Koefisien Model : [[1.18109091]]
Intercept Model : [-4.12617727]
```

Pada uji coba, seperti yang telah dijelaskan bahwa algoritma ini akan memberikan luaran berupa probabilitas. Kotak merah merupakan proabilitas untuk label 0, sedangkan Biru untuk label 1

```
model.predict_proba(x)
```

```
array([[0.98411203, 0.01588797],
       [0.95003074, 0.04996926],
       [0.85370936, 0.14629064],
       [0.64173546, 0.35826454],
       [0.35475873, 0.64524127],
       [0.1443924 , 0.8556076 ],
       [0.04924876, 0.95075124],
       [0.01565079, 0.98434921],
       [0.00485659, 0.99514341],
       [0.00149573, 0.99850427]])
```

Adapun hasil prediksi dan akurasi adalah sebagai berikut

```
model.predict(x)
```

```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

```
model.score(x, y)
```

```
1.0
```

6.4 Studi Kasus Pendeteksian Penyakit Diabetes

Dataset dalam studi kasus ini adalah dataset PIMA Indians diabetes. Data ini diperoleh dari UCI yang aslinya dimiliki oleh *National Institute of Diabetes and Digestive and Kidney Diseases* sebagai dataset yang diberikan secara bebas kepada siapa saja yang ingin melakukan penelitian. Dataset Pima ini terdiri dari 768 data klinis yang semuanya berasal dari jenis kelamin wanita dengan umur sekurang - kurangnya 21 tahun. Data ini memiliki 8 atribut dengan target output positif diabetes (ditunjukkan dengan output = 1) dan negatif diabetes (ditunjukkan dengan output = 0). Daftar atribut data diabetes ditunjukkan pada Tabel 1

Keterangan	Satuan	Nama Attribut	Tipedata
Jumlah kehamilan	-	jHamil	Numerik
Kadar glukosa	mg/dl	jGlukosa	Numerik
Tekanan darah diastolic	mm Hg	tekananDarah	Numerik
Ketebalan kulit pada trisep	Mm	ketebalanKulit	Numerik
Serum insulin	mu U/ml	serumInsulin	Numerik
Massa indeks tubuh	kg/ m^2	IMB	Numerik
Fungsi riwayat diabetes	-	RiyawatDiabetes	Numerik
Umur	Tahun	Umur	Numerik
Positif diabetes	Positif Diabetes(1) Negatif Diabetes(0)	Class / Label	Nominal

Langkah pertama adalah memuat data kedalam Jupyter notebok

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv("diabetes.csv")
df.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Adapun tahap-tahap penyelesaian masalah adalah

6.4.1 Preprocessing

Sebelum dilakukan proses klasifikasi, data harus dipersiapkan terlebih dahulu agar siap untuk diolah (dikenal dengan istilah pre-processing) dengan tujuan meminimalkan kesalahan dan mengoptimalkan model yang akan dihasilkan. Adapun tahap-tahap persiapan yang dilakukan adalah:

1. **Penanganan data yang tidak lengkap (*Missing Value)***

Berdasarkan info datset, tidak terdapat data yang hilang. Semua kolom memiliki jumlah item yang sama

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
df.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Namun jika diperhatikan ada nilai yang aneh pada skintickness dan insulin dimana ada yang bernilai 0. Nilai nol tersebut tidak mungkin karena insulin pasti nilainya diatas 0. Setelah ditelusuri, ada 5 kolom yang bernilai 0 diantaranya 'Glucose','BloodPressure','SkinThickness','Insulin', dan 'BMI'. Langkah selanjutnya adalah mengganti nilai 0 sebagai nilai yang hilang.

```
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] =  df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
df.isnull().sum()
```

```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Dari hasil analisa dataset Pima Indians dapat diketahui bahwa tidak semua atribut memiliki nilai yang lengkap, dimana kelengkapan atribut ini akan menentukan seberapa baik hasil dari pengklasifikasi. Jumlah data tidak lengkap pada masing–masing atribut yaitu atribut jumlah glukosa sebanyak 5, atribut tekanan darah sebanyak 35, atribut ketebalan trisep sebanyak 227, atribut insulin sebanyak 374, atribut IMB sebanyak 11, sedangkan atribut umur dan kelas memiliki nilai yang lengkap. Atribut yang memiliki *missing value* akan dilakukan proses imputation menggunakan pendekatan median

```
df['Glucose'] = df['Glucose'].fillna(df['Glucose'].median())
df['BloodPressure'] = df['BloodPressure'].fillna(df['BloodPressure'].median())
df['SkinThickness'] = df['SkinThickness'].fillna(df['SkinThickness'].median())
df['Insulin'] = df['Insulin'].fillna(df['Insulin'].median())
df['BMI'] = df['BMI'].fillna(df['BMI'].median())

```

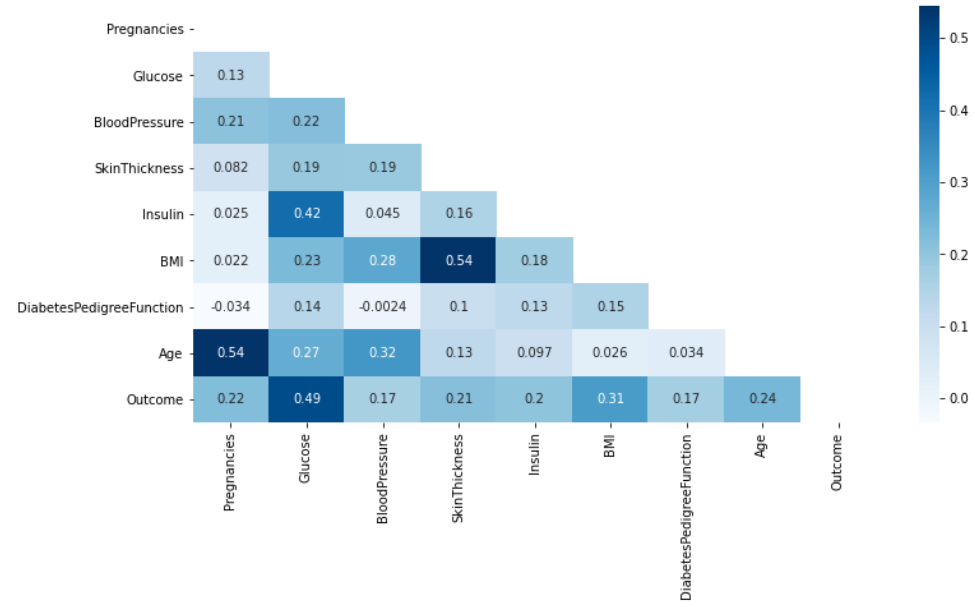
2. Analisa Korelasi

Analisis korelasi adalah teknik analisis yang digunakan untuk mengukur kuat lemahnya hubungan dua variabel. Variabel ini terdiri dari variabel bebas dan tergantung. Besarnya hubungan berkisar antara 0-1. Pada

penelitian ini menggunakan korelasi pearson. Korelasi pearson digunakan untuk mengetahui ada tidaknya hubungan antara 2 variabel, yaitu variabel bebas dan variabel tergantung yang berskala interval atau rasio (parametrik).

```
1  plt.figure(figsize=(12,6))
2  matrix = np.triu(df.corr())
3  sns.heatmap(df.corr(), annot=True, cmap='Blues', mask=matrix)
```

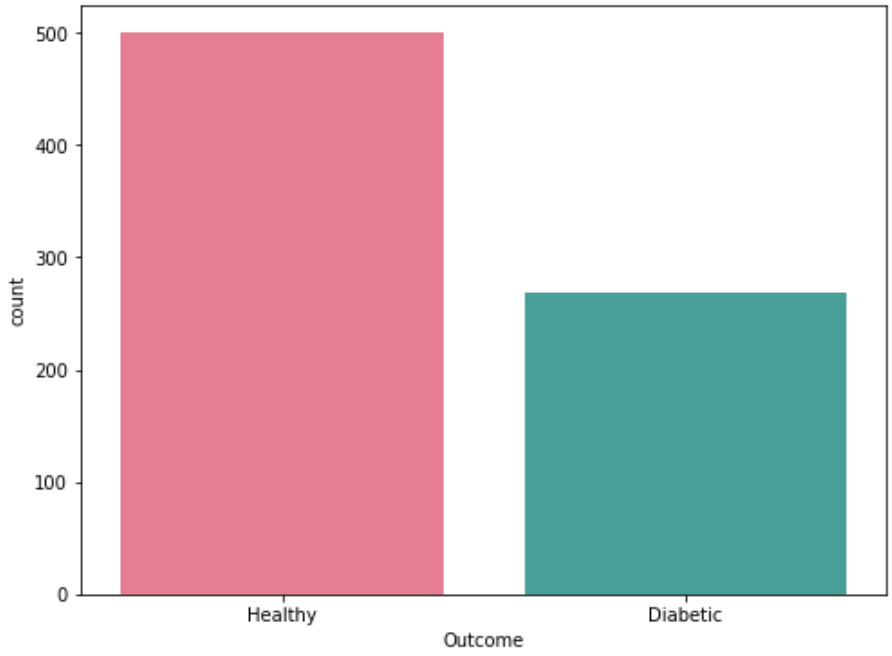
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb3e30a91d0>
```



Atribut (pregnancy)kehamilan dan umur (age) serta BMI dan Skin Tickness memiliki korelasi yang cukup kuat dengan nilai 0.54 artinya lebih kurang 50% atribut tersebut berkolerasi.

```
from matplotlib.pyplot import figure, show

figure(figsize=(8,6))
ax = sns.countplot(x=df['Outcome'], data=df,palette="husl")
ax.set_xticklabels(["Healthy","Diabetic"])
healthy, diabetics = df['Outcome'].value_counts().values
print("Jumlah diabetic: ", diabetics)
print("Jumlah healthy : ", healthy)
```

```
Jumlah diabetic:  268
Jumlah healthy :  500
```



6.4.2 Hasil Pelatihan dan Pengujian

```
# Pecah daya menjadi train dan test
from sklearn.model_selection import train_test_split
X = df.drop('Outcome', axis=1)
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
logmodel = LogisticRegression(max_iter=200)
logmodel.fit(X_train, y_train)
prediction1 = logmodel.predict(X_test)
```

```
print('Confusion Matrix:\n', confusion_matrix(y_test, prediction1))
print('\n')
print('Classification Report:\n', classification_report(y_test, prediction1))
```

```
Confusion Matrix:
 [[124  27]
 [ 33  47]]


Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.82      0.81       151
           1       0.64      0.59      0.61        80

    accuracy                           0.74       231
   macro avg       0.71      0.70      0.71       231
weighted avg       0.74      0.74      0.74       231
```

Prediksi	Referensi	
	Negatif	Positif
Negatif	124	27
Positif	33	74

Bab

7

Naive Bayes

Orang yang berdoa tanpa beramal sama halnya seperti pemanah tanpa busur. – Ali bin Abi Thalib

Teorema Bayes (atau hukum Bayes atau aturan Bayes) memainkan peranan yang sangat penting bidang Machine Learning. Teori ini pertama kali dikembangkan oleh Thomas bayes (1702-1763), tetapi beliau tidak pernah mempublikasikan idenya tersebut, bahkan sampai pertengahan abad ke-18 masih belum dikenal istilah "Probabilitas". Istilah yang muncul pertama kali adalah "*Doctrine of Changes*" – diambil dari sebuah buku karangan Abraham de Moievre, seorang ahli matematika dari prancis. Sebuah artikel berjudul "*An Essay towards solving a Problem in the Doctrine of Chances*", pertama ditulis oleh Bayes, diedit dan diubah oleh temannya Richard Price lalu dibacakan untuk Royal Society dan diterbitkan dalam Philosophical Transactions of the *Royal Society of London* tahun 1763. Dalam esai ini, Bayes menjelaskan sebuah teorema sederhana tentang probabilitas gabungan yang menimbulkan perhitungan Probabilitas Inverse (Teorema bayes).

Gambar 7.1. Thomas Bayes (1701 – April 1761) (sumber:wikipedia.com)

Teorema Bayes merupakan kaidah yang memperbaiki atau merevisi suatu probabilitas dengan cara memanfaatkan informasi tambahan. Maksudnya, dari probabilitas awal (*prior probability*) yang belum diperbaiki dengan

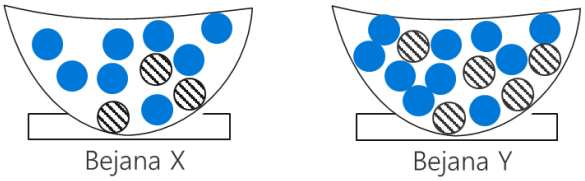
rumuskan berdasarkan informasi yang tersedia saat ini, kemudian dibentuklah probabilitas berikutnya (*posterior probability*) digunakan unruk menjelaskan peluang suatu peristiwa, berdasarkan pengetahuan tentang kondisi peluang yang terkait dengan peristiwa tersebut. Dalam penafsiran Bayes, teorema ini menyatakan seberapa jauh derajat kepercayaan subjektif harus berubah secara rasional ketika ada petunjuk baru.

Sebagai contoh Budi berbicara dengan seseorang didalam kelas. Dari informasi ini kita dapat mengetahui peluang budi berbicara dengan perempuan adalah 50% (*prior probability)* karena hanya ada dua peluang yaitu berbicara dengan laki-laki atau perempuan. Ketika didatangkan informasi tambahan bahwa orang yang berbicara dengan budi memiliki rambut yang panjang (*posterior probability*), maka peluang budi berbicara dengan perempuan berubah menjadi lebih besar. Hal tersebut terjadi karena peluang seorang wanita untuk memiliki rambut panjang lebih besar dari laki-laki berambut panjang. Akibatnya peluang budi berbicara dengan seorang wanita berubah karena ada informasi tambahan. Fenomena inilah yang ingin dijelaskan menggunkan teorema bayes.

7.1 Terorema Bayes

Dalam tulisannya yang diterbitkan tahun 1763, 3 tahun setelah kematiannya, Bayes memperkenalkan sebuah versi dari persamaan beberapa probabilitas yang sekarang dikenal sebagai teorema Bayes. Saat paper ini pertama kali terbit, hanya ada sedikit ekspektasi bahwa persamaan sederhana ini bisa memecahkan banyak permasalahan dalam teori peluang. Namun siapa sangka jika dua ratus tahun kemudian, teorema Bayes telah menjadi sesuatu yang penting dan saat ini menjadi dasar bagi inferensi statistik Bayesian. Untuk memahami teorema Bayes, kita harus pahami dulu peluang bersyarat, maka perhatikan kasus berikut ini.

Misal anda memiliki dua bejana berisi bola yang berwarna jingga dan biru. Pada kasus ini, anda mengetahui berapa jumlah bola jingga dan biru pada masing masing bejana, sehingga dengan mudah anda dapat menghitung berapa probabilitas terambilnya bola biru dari bejana pertama.



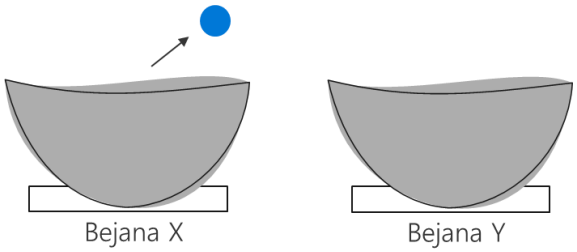
Gambar 7.2. Bejana X dan Y berisi bola biru dan jingga

Anda dapat dengan mudah menghitung peluang terambilnya bola jingga pada bejana X menggunakan persamaan 1. Karena jumlah semua bola di bejana X adalah 11 dan jumlah bola jingga adalah 3 maka peluang terambilnya bola jingga adalah 3/11, P(bola=jingga)=3/11.

DEFINISI PENTING

Peluang atau probabilitas adalah sebuah nilai yang dapat digunakan untuk mengungkapkan pengetahuan atau tingkat kepercayaan bahwa suatu kejadian akan berlaku atau telah terjadi

Namun jika saya mengambil sebuah bola dari salah satu bejana secara acak, lalu saya mendapatkan sebuah bola biru, maka dapatkah anda menentukan peluang bahwa bola tersesebut terambil dari bejana X? pertanyaan ini dapat dijawab dengan menggunakan teorema bayes.

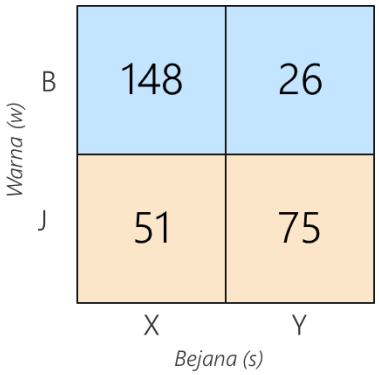


Gambar 7.3. Bejana X dan Y berisi bola biru dan jingga

Untuk menjawab kasus ini, kita dapat melakukan serangkaian percobaan pengambilan bola secara acak. Untuk itu saya membutuhkan bantuan sebuah koin untuk memilih pengambilan dari bejana X atau Y. Jika koin saya lempar lalu menghasilkan gambar, maka saya akan mengambil bola dari bejana X, jika angka maka saya akan mengambil dari bejana Y. Percobaan ini saya akan melakukan percobaan sebanyak 300 kali. Adapun variabel yang kita observasi adalah bejana (s) dan warna bola (w). Agar mempermudah maka saya akan menyingkat Bola Jingga dengan J, Bola Biru dengan B, Bejana X dengan X dan Bejana Y dengan Y sehingga masing-masing variabel dapat ditulis $s=\{X,Y\}$ dan $w=\{J,B\}$.

Dari hasil percobaan didapatkan data sebagai berikut

- Jumlah bola biru yang terambil dari bejana X berjumlah 148, n(s=X, w=B) = 148.
- Jumlah bola jingga yang terambil dari bejana X berjumlah 51, n(s=X, w=J) = 51.
- Jumlah bola biru yang terambil dari bejana Y berjumlah 26, n(s=Y, w=B) = 26.
- Jumlah bola jingga yang terambil dari bejana Y berjumlah 75, n(s=Y, w=J) = 75



Berdasarkan informasi-informasi tersebut maka kita dapat menjawab beberapa pertanyaan berikut ini:

P1. Berapa peluang terambil bola dari bejana X?

Pertanyaan tersebut dapat ditulis sebagai $p(s = X)$. Peluang tersebut dihitung dengan menjumlahkan semua nilai yang terambil dari bejana X . Proses perhitungannya sebagai sebagai berikut:

$$p(s = X) = \frac{n(s = X, w = B) + n(s = X, w = J)}{N} \quad (7\text{-}1)$$

$$p(s = X) = \frac{148 + 51}{300} \approx \frac{2}{3}$$

Dan peluang bola terambil dari bejana Y adalah

$$p(s = Y) = 1 - p(s = X) \approx \frac{1}{3}$$

P2. Berapa peluang terambil bola berwarna biru (B)?

Pertanyaan ini dapat ditulis sebagai $p(w = B)$. Peluang tersebut dihitung dengan menjumlahkan semua bola yang terambil berwarna biru. Proses perhitungannya sebagai sebagai berikut:

$$p(w = B) = \frac{n(s = X, w = B) + n(s = Y, w = B)}{N} \quad (7\text{-}2)$$

$$p(w = B) = \frac{148 + 26}{300} = \frac{174}{300} \approx \frac{6}{10}$$

Dan peluang bola terambil dari bejana Y adalah

$$p(w = J) = 1 - p(w = B) \approx \frac{4}{10}$$

Probabilitas yang seperti $p(s = X)$, $p(s = Y)$, $p(w = B)$, dan $p(w = J)$ dinamakan *prior probability.* Prior Probability adalah peluang sebelum data baru dikumpulkan.

P4. Berapakah peluang terambil bola biru pada bejana X?

Pertanyaan ini adalah contoh dari Join Probability dimana probabilitas event 1 (bola biru terambil) dan event 2 (terambil dari bejana X) terjadi pada saat yang sama. Untuk menghitungnya maka perlu diketahui jumlah $n(w = B, s = X)$.

$$p(s = X, w = B) = \frac{n(s = X, w = B)}{N} = \frac{148}{300} \approx \frac{1}{2}$$

P5. Berapakah peluang bola yang terambil adalah biru jika kita mendapatkan bola dari bejana X?

Pertanyaan ini menarik karena ada kondisi yang telah terpenuhi yaitu kita telah mengambil dari bejana X. Probalitas yang seperti ini biasa disebut *conditional probability*. Pada kasus ini kita telah mengetahui bola tersebut berasal dari X. Berdasarkan pengetahuan tersebut kita akan menghitung sebuah probabilitas yang menggambarkan likelihood pengambilan bola biru.

Probabilitas ini dapat ditulis $p(w = B|s = X)$, dimana $s = X$ merupakan fakta atau kondisi yang telah terpenuhi. Untuk menghitungnya kita perlu mengetahui jumlah bola biru B yang terambil dari X

$$p(w = B|s = X) = \frac{n(w = B, s = X)}{n(w = B, s = X) + n(w = J, s = X)}$$

$$p(w = B|s = X) = \frac{148}{148 + 51} = \frac{148}{199} \approx \frac{3}{4}$$

Apabila kita review persamaan pada kasus peluang terambil bola biru (P5) pada bejana X, $p(w = B, s = X)$, lalu kita kembangkan dengan mengalikannya dengan $n(w = B, s = X) + n(w = O, s = X)$ pada pembilang dan penyebut maka

$$p(w = B, s = X) = \frac{n(w = B, s = X)}{N} \cdot \frac{n(w = B, s = X) + n(w = O, s = X)}{n(w = B, s = X) + n(w = O, s = X)}$$

$$p(s = X, w = B,) = \ p(w = B|s = X) \cdot p(s = X)$$

Relasi antara peluang ini disebut *product rule*. Dengan *product rule* memungkinkan kita untuk menhitung *joint probability* p(w=B, s=X) menggunakan *conditional probability* p(w=B| s=X) dan *prior probability* p(s=X).

Berdasarkan product rule itu dapat diturunkan menjadi teorema bayes dimana p(w, s) bernilai sama dengan p(w, s) maka

$$p(w, s) = p(s|w)p(w)$$

$$p(s, w) = p(w|s)p(s)$$

$$p(w|s) = \frac{p(s|w)p(w)}{p(s)} \qquad (7\text{-}3)$$

Dengan menggunakan teorama bayes tersebut, maka kita dapat menjawab pertanyaan, dari bejana mana sebuah bola biru diambil? Teorema Bayes memungkinkan kita untung menghitung *conditional probability* untuk menjawab pertanyaan tersebut.

Pada kasus tersebut yang menjadi fakta adalah bola yang diambil berwarna biru, w=B. Jawabnya dapat ditentukan dengan menghitung p(s|w=B) untuk s=X dan s=Y, lalu kita pilih peluang yang paling besar sebagai jawaban.

Kemungkinan 1. S=X

$$p(s = X|w = B) = \frac{p(w = B|x = X)p(s = X)}{p(w = B)}$$

Subtitusi $p(w = B)$ dengan $p(s = X, w = B) + p(s = Y, w = B)$

$$p(s = X|w = B) = \frac{p(w = B|x = X)p(s = X)}{p(s = X, w = B) + p(s = Y, w = B)}$$

$$p(s = X|w = B) = \frac{{}^{3}/_{4} \cdot {}^{2}/_{3}}{{}^{1}/_{2} + {}^{1}/_{12}} \approx 0.86$$

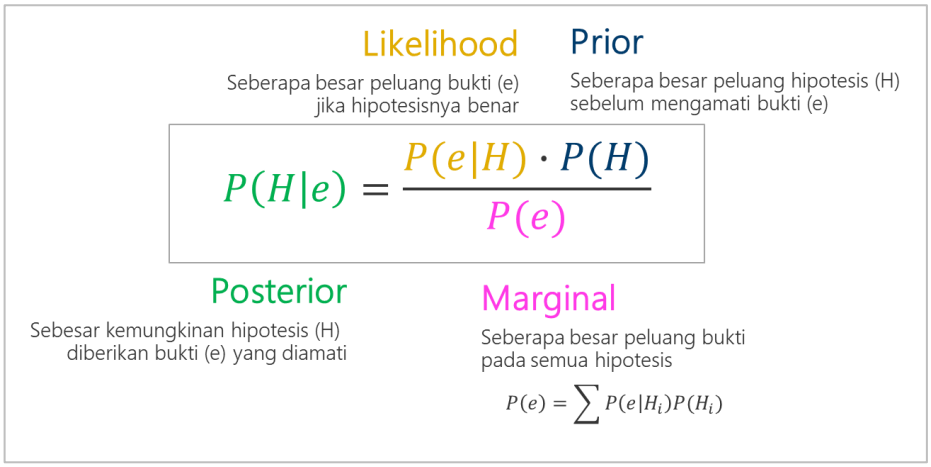
Kemungkinan 2. S=Y

$$p(s = Y|w = B) = \frac{p(w = B|x = Y)p(s = Y)}{p(s = X, w = B) + p(s = Y, w = B)}$$

$$p(s = X|w = B) = \frac{{}^{1}/_{4} \cdot {}^{1}/_{3}}{{}^{1}/_{2} + {}^{1}/_{12}} \approx 0.14$$

Jadi kemungkinan bola biru terambil dari bejana X sebesar 86%, jadi kemungkinan besar bola tersebut dari bejana X.

Secara umum, formula *teorema bayes* dinyatakan sebagai berikut



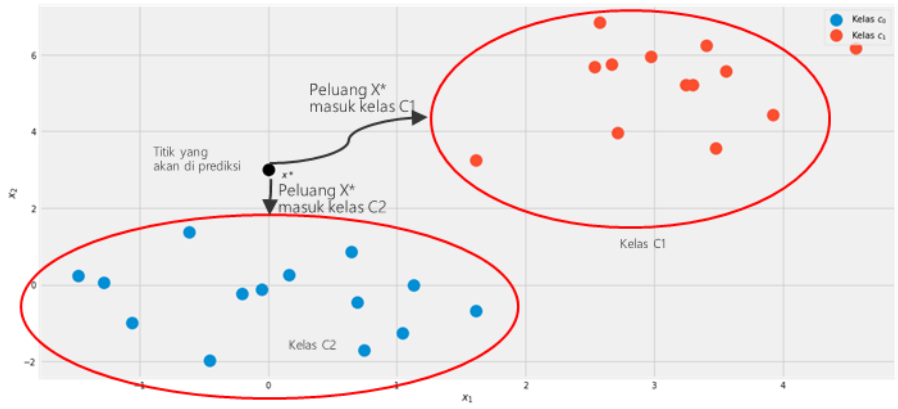
Gambar 7.4. Formula Theorem Bayes

Sebagai contoh anda sedang memegang kulit kerang, seberapa besar peluang anda dekat dengan lautan?. Maka e atau bukti adalah anda memegang kerang dan Hipotesisnya adalah anda dekat dengan lautan maka perumusan Bayesnya adalah (Peluang saya memegang kulit kerang ketika di dekat laut) x (Peluang saya di dekat laut) / (Peluang saya memegang kulit kerang)

$$P(\text{Saya didekat Laut} \mid \text{Saya memegang Kulit Kerang}) = \frac{P(\text{Saya memegang Kulit Kerang} \mid \text{Saya didekat Laut})\, P(\text{Saya didekat Laut})}{P(\text{Saya memegang Kulit Kerang})}$$

7.2 Naïve Bayes Clasifier

Naïve Bayes Classifer (NBC) merupakan sebuah teknik yang memanfaatkan teorema bayes untuk melakukan klasifikasi terhadap data. NBC berasumsi bahwa semua fitur pada data bersifat independen. Cara kerjanya sangat sederhana, bayangkan ada memiliki data kucing dan anjing yang telah di plot dalam grafik berikut ini



Sebagai contoh sebuah mobil dapat diklasifikasikan berdasarkan Warna, CC, dan Harga. Ketika menggunakan NBC maka kita berasumsi Warna, CC, dan Harga bersifat independen, oleh karena itu algoritma ini dinamakan "Naïve".

Algoritma NBC sangat sederhana dan memiliki performa yang cukup baik dibandingkan algoritma yang lain. Misal kita ingin mengklasifikasikan keputusan untuk bermain atau tidak berdasarkan sebuah dataset yang memiliki 4 fitur (*outlook,temperatur, humidity, windy*) dan 1 label (play).

Outlook	Temperature	Humidity	Windy	Play
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes

Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Berdasarkan tabel tersebut jika hari ini Outlook=Sunny, Temperature=Normal, Humidity=High dan Windy= False, apa keputusan anda Play=Yes atau No?

Adapun langkah-langkah algoritma ini adalah sebagai berikut

1. **Membuat tabel Frekwensi dan likelihood dari semua fitur pada data training.**

   pada kasus ini kita akan membuat 4 tabel frekwensi dan likelihood yaitu Outlook, Temperature, Humidity dan wind. Dan 1 tabel frekwensi dan peluang label.

Likelihood

Outlook				
	Yes	No	P(yes)	P(No)
Sunny	3	2	3/9	2/5
Overcast	4	0	4/9	0/5
Rainy	2	3	2/9	3/5
Total	9	5	-	-

Temperature				
	Yes	No	P(yes)	P(No)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	-	-

Humidity				
	Yes	No	P(yes)	P(No)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	-	-

Wind				
	Yes	No	P(yes)	P(No)
FALSE	6	2	6/9	2/5
TRUE	3	3	3/9	3/5
Total	9	5	-	-

Prior

Play	N	P
Yes	9	9/14
No	5	5/14
Total	14	

2. **Menggunakan formula naïve bayes untuk menghitung posterior untuk setiap kelas.**

   Pada kasus ini maka evidence/buktinya adalah Outlook=Sunny, Temperature=Mild, Humidity=High dan Windy= False. Sedangkan hipotesisnya adalah Play=Yes dan Play=No. Maka kita akan menghitung 2 Posterior yaitu

   - P(**Play=Yes** | Outlook=Sunny, Temperature=Normal, Humidity=High, Windy= False)
   - P(**Play=No** | Outlook=Sunny, Temperature= Mild, Humidity=High , Windy= False)

   Untuk kasus multi-eviden, maka persamaan bayes yang digunakan adalah sebagai berikut:

$$P(y|x_1, x_2, .., x_n) = \frac{p(x_1|y) \cdot p(x_2|y) \dots p(x_2|y) \cdot p(y)}{p(x_1)p(x_2) \dots p(x_n)}$$

Jika kita sederhanakan maka

$$P(H|x_1, x_2, .., x_n) = \frac{p(H)}{p(x_1)p(x_2) \dots p(x_n)} \prod_{i=1}^{n} p(x_i|y)$$

Jika kita perhatikan formula diatas, penyebut $p(x_1)p(x_2) \dots p(x_n)$ bersifat konstan, artinya tidak berpengaruh terhadap nilai, sehingga dapat diabaikan dalam proses pengambilan keputusan agar mempermudah komputasi.

$$P(y|x_1, x_2, .., x_n) \propto p(y) \prod_{i=1}^{n} p(x_i|y)$$

- P(**Play=Yes** | Outlook=Sunny, Temperature= Mild, Humidity=High, Windy= False) ∝ P(Outlook=Sunny| **Play=Yes)** P(Temperature= Mild | **Play=Yes)** P(Humidity=High | **Play=Yes)** P(Windy= False | **Play=Yes)**

  Berdasarkan tabel pada langkah 1, diketahui

  - P(**Play=Yes**) = $\frac{9}{14}$
  - P(Outlook=Sunny| **Play=Yes)**= $\frac{2}{9}$
  - P(Temperature= Mild | **Play=Yes)** = $\frac{4}{9}$
  - P(Humidity=High | **Play=Yes)** = $\frac{3}{9}$
  - P(Windy= False | **Play=Yes)** = $\frac{6}{9}$

  Maka,

  P(**Play=Yes** | Outlook=Sunny, Temperature= Mild, Humidity=High, Windy= False) = $\frac{9}{14}\left[\frac{3}{9}\frac{4}{9}\frac{3}{9}\frac{6}{9}\right]$=$\frac{1944}{91854} = 0.0211$

- P(**Play=No** | Outlook=Sunny, Temperature= Mild, Humidity=High, Windy= False) ∝ P(Outlook=Sunny| **Play= No)**

P(Temperature= Mild | **Play= No)** P(Humidity=High | **Play= No)** P(Windy= False | **Play= No)**

- P(**Play= No**) = $\frac{5}{14}$
- P(Outlook=Sunny| **Play= No)**= $\frac{3}{5}$
- P(Temperature= Mild | **Play= No)** = $\frac{2}{5}$
- P(Humidity=High | **Play= No)** = $\frac{4}{5}$
- P(Windy= False | **Play= No)** = $\frac{2}{5}$

Maka,

P(**Play= No** | Outlook=Sunny, Temperature= Mild, Humidity=High, Windy= False) = $\frac{5}{14}\left[\frac{2}{5}\frac{2}{5}\frac{4}{5}\frac{2}{5}\right]$=$\frac{160}{8750} = 0.0182$

3. Membandingkan semua posterior, lalu memilih posterior yang terbesar.

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg \max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Berdasarkan hasil perhitungan, probabilitas hipotest Play=No lebih besar dari Play=No sehingga data Outlook=Sunny, Temperature=Mild, Humidity=High dan Windy= False diklasifikasi sebagai NO.

Implementasi Naïve Bayes dari awal (tanpa library)

NBC merupakan algoritma yang cukup sederhana, kita dapat mengimplementasikannya dengan mudah, kita hanya perlu membuat method untuk menghitung likelihood dan prior probability pada proses training. Implementasi dapat dilihat pada lampiran source code "`7.Bayes/naïve_bayes.py`".

Pada sourcecode 7.1, proses training dilakukan pada baris 18 sampai 30. Pada training, ada dua proses utama yang dilakukan yaitu menghitung likelihood dan prior (baris 29 dan 30). Sebelum melakukan itu, kita perlu mendapatkan fitur-fitur data (baris 20), menyimpan data training dan label (baris 21,22), menyimpan jumlah data train dan jumlah fitur (baris 23,24). Setelah itu semua likelihood dan prior perlu diberi nilai awal yaitu nol (baris 26)

7.1 Naïve Bayes tanpa library

```
1  import os
2  import numpy as np
3  import pandas as pd
4
5  class NaiveBayes:
6    def __init__(self):
7
8      self.features = list
9      self.likelihoods = {}
10     self.class_priors = {}
11
12     self.X_train = np.array
13     self.y_train = np.array
14     self.train_size = int
15     self.num_feats = int
16     self.debug = True
17
18   def fit(self, X, y):
19
20     self.features = list(X.columns)
21     self.X_train = X
22     self.y_train = y
23     self.train_size = X.shape[0]
24     self.num_feats = X.shape[1]
25     #setting semua nilai likelihood dan prior menjadi 0
```

```
26        self._init_value()
27
28        #step 1. hitung likelihood dan prior
29        self._hitung_likelihood()
30        self._hitung_class_prior()
31
32    def _init_value(self):
33        for feature in self.features:
34            self.likelihoods[feature] = {}
35            for feat_val in np.unique(self.X_train[feature]):
36                for outcome in np.unique(self.y_train):
37                    self.likelihoods[feature]
38                        .update({feat_val+'_'+outcome: 0})
39                    self.class_priors.update({outcome: 0})
40
41    def _hitung_class_prior(self):
42        for outcome in np.unique(self.y_train):
43            outcome_count = sum(self.y_train == outcome)
44            if self.debug:
45                print("Prior (", outcome, ") = ",
46                      outcome_count, "/", self.train_size)
47            self.class_priors[outcome]
48                = outcome_count / self.train_size
49
50    def _hitung_likelihood(self):
51        for feature in self.features:
52            for outcome in np.unique(self.y_train):
53                outcome_count = sum(self.y_train == outcome)
54                feat_likelihood = self.X_train[feature]
55                  [self.y_train[self.y_train == outcome]
56                  .index.values.tolist()].value_counts().to_dict()
57
58                for feat_val, count in feat_likelihood.items():
59                    if (self.debug):
60                        print('Likelihood(',feature, '=', feat_val,
61                              '|', outcome, ") :",
62                              count, '/', outcome_count)
63                    self.likelihoods[feature]
64                          [feat_val + '_' + outcome]
65                           = count/outcome_count
66
67
68    def predict(self, X):
69        results = []
70        X = np.array(X)
71
```

```
72      for query in X:
73        probs_outcome = {}
74        for outcome in np.unique(self.y_train):
75          prior = self.class_priors[outcome]
76          likelihood = 1
77
78          for feat, feat_val in zip(self.features, query):
79            likelihood *= self.likelihoods[feat]
80                  [feat_val + '_' + outcome]
81
82          posterior = (likelihood * prior)
83
84          probs_outcome[outcome] = posterior
85          if self.debug:
86            print("Posterior(", outcome, "|",query,") :"
87                        , posterior)
88
89        result = max(probs_outcome,
90                     key=lambda x: probs_outcome[x])
91        results.append(result)
92
93      return np.array(results)
94
```

7.3 Gaussian Naïve Bayes

Gaussian Naïve Bayes merupakan salah satu varian dari Naïve Bayes. Prinsip dasarnya sama dengan Naïve Bayes, namun pada Naïve Bayes kita tidak dapat menghitung probabilitas data continues. Oleh karena itu kita menggunakan distribusi gaussian untuk menghitung probabilitas dengan asumsi data terdistribusi normal. Adapun pdf (*probability density function*) dari gaussian distribution adalah

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Metode ini cocok digunakan pada data yang bersifat continues dan berdistribusi normal. Contohnya pendeteksian transaksi keuangan yang aneh dimana data transaksi keuangan terdistribusi normal.

Sebagai contoh, misalnya kita memiliki 40 data yang direpresentasikan oleh titik-titik yang tersebut dibagi menjadi tiga kelas yaitu C1, C2 dan C2. Dengan menggunakan Gaussian kita akan mencoba menemukan mean dan varian dari tiap-tiap kelas. Setelah menemukan mean dan varian tersebut kita akan mencoba memperdiksi titik x* (-2,5) termasuk dalam kelas apa.

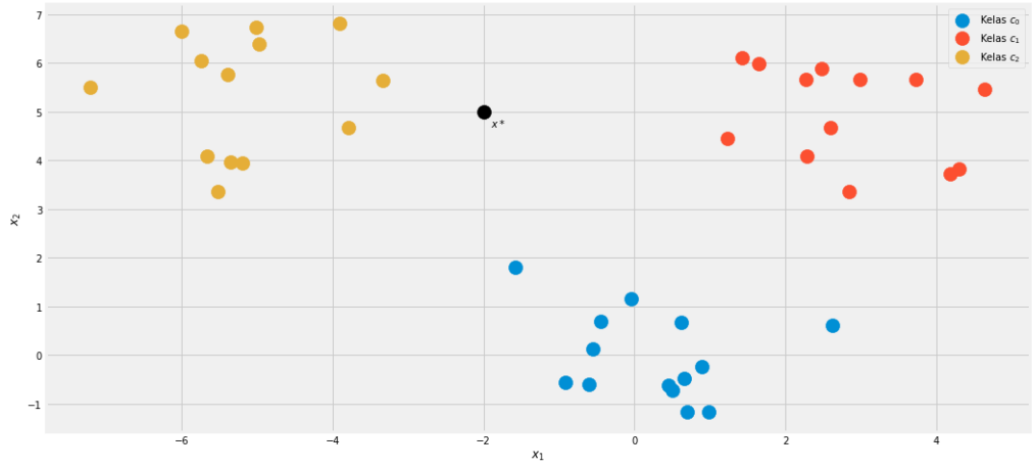
Langkah pertama meload library yang dibutuhkan:

```
1  import numpy as np
2  from sklearn.datasets import make_blobs
3  import matplotlib.pyplot as plt
4  plt.style.use('fivethirtyeight')
```

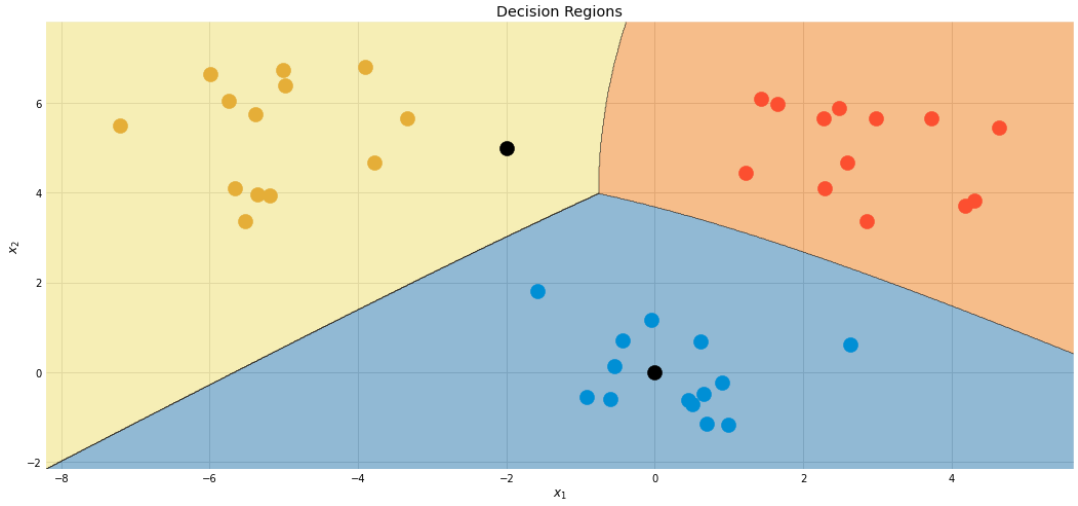
Lalu, membuat dataset dan menampilkan nya

```
#membuat dataset
X, y = make_blobs(n_samples=40, centers=[(0,0), (3,5), (-5, 5)], random_state=86)

plt.figure(figsize=(16, 8))
for i in range(3):
    plt.scatter(X[np.where(y==i), 0], X[np.where(y==i), 1], s=200, label=f'Kelas $c_{i}$')
plt.scatter([-2], [5], c='k', s=200)
plt.annotate('$x*$', (-1.9, 4.7))
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f5e419ab890>
```



Adapun tampilan area keputusannya adalah



7.4 Studi Kasus – Prediksi Income

Dataset ini berasal dari data sensus di Amerika Serikat tentang pendapatan orang dewasa pada tahun 1994. Dataset ini berisi 32.561 data dengan kolom

- Umur peserta sensus
- Kelompok pekerjaan: berisi status kepegawaian seperti Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
- Final Weight
- Pendidikan: berisi Pendidikan tertinggi seseorang dengan nilai Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- Nomer Pendidikan. berisi Pendidikan tertinggi seseorang dengan nilai numerik
- Status Perkawinan
- Pekerjaan: berisi Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- Relationship: berisi hubungan dengan invidu lainnya seperti Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- Ras berisi White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
- Kelamin
- Capital gain
- Capital Loss
- Hour Perweek
- Negara
- Income (label)

7.4.1 Load Dataset

Langkah pertama adalah membaca file income.csv. file ini tidak memiliki header dan dibatasi oleh ",[spasi]". Dengan menggunakan pandas, kita akan menyimpan dalam bentuk dataframe.

```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv("income.csv", header=None,sep=',\s')
df.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does
  """Entry point for launching an IPython kernel.
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Agar mempermudah dalam identifikasi kolom, maka kolom-kolom tersebut akan diberi nama sesuai dengan keterangan pada dataset.

```
#memberikan header pada data
col_names = ['umur', 'kategori_pekerjaan', 'final_weight', 'pendidikan', 'jumlah_pendidikan', 'status_pernikahan', 'pekerjaan',
             'relationship',  'ras', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'negara', 'income']
df.columns = col_names
df.columns
```

```
Index(['umur', 'kategori_pekerjaan', 'final_weight', 'pendidikan',
       'jumlah_pendidikan', 'status_pernikahan', 'pekerjaan', 'relationship',
       'ras', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week',
       'negara', 'income'],
      dtype='object')
```

Setelah itu, kita akan mellihat tipe data dan jumlah data yang ada pada dataset. Sekilas, tidak ada permasalahan terhadap data.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   umur                32561 non-null  int64
 1   kategori_pekerjaan  32561 non-null  object
 2   final_weight        32561 non-null  int64
 3   pendidikan          32561 non-null  object
 4   jumlah_pendidikan   32561 non-null  int64
 5   status_pernikahan   32561 non-null  object
 6   pekerjaan           32561 non-null  object
 7   relationship        32561 non-null  object
 8   ras                 32561 non-null  object
 9   sex                 32561 non-null  object
 10  capital_gain        32561 non-null  int64
 11  capital_loss        32561 non-null  int64
 12  hours_per_week      32561 non-null  int64
 13  negara              32561 non-null  object
 14  income              32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

7.4.2 Data Preprocessing

Selanjutnya penulis akan mengispeksi data-data kategorikal seperti kategori_pekerjaan, Pendidikan, status_pernikahan dan lain-lain.

```
# Cari file kategorial
kolom_kategorical = ['kategori_pekerjaan', 'pendidikan', 'status_pernikahan', 'pekerjaan', 'relationship', 'ras', 'sex', 'negara', 'income']
df[kolom_kategorical].head()
```

	kategori_pekerjaan	pendidikan	status_pernikahan	pekerjaan	relationship	ras	sex	negara	income
0	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	United-States	<=50K
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States	<=50K
2	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States	<=50K
3	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	United-States	<=50K
4	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	Cuba	<=50K

Semua data kategorikal tersebut tidak memiliki NULL atau nilai kosong

```
df[kolom_kategorical].isnull().sum()
```

```
kategori_pekerjaan    0
pendidikan            0
status_pernikahan     0
pekerjaan             0
relationship          0
ras                   0
sex                   0
negara                0
income                0
dtype: int64
```

Nampun apabila ditampilkan nilai uniknya, kolom kategori_pendidikan, pekerjaan dan negara memiliki nilai " ? " yang berarti kosong

```
for var in kolom_kategorical:
  print(df[var].value_counts())
```

```
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: pekerjaan, dtype: int64
Husband           13193
Not-in-family      8305
Own-child          5068
Unmarried          3446
Wife               1568
Other-relative      981
Name: relationship, dtype: int64
White                27816
Black                 3124
Asian-Pac-Islander    1039
Amer-Indian-Eskimo     311
Other                  271
Name: ras, dtype: int64
Male      21790
Female    10771
Name: sex, dtype: int64
United-States                 29170
Mexico                          643
?                               583
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
```

Lalu penulis mengganti nilai "?" menjadi null, sehingga penulis dapat melihat berapa jumlah data yang kosong dan mudah dalam memperbaikinya

```
for var in kolom_kategorical:
  df[var].replace("?", np.NaN, inplace=True)

```

Terdapat 1836 kategori Pendidikan, 1843 pekerjaan dan 583 negara yang kosong. Karena jumlahnya cukup banyak maka kita tidak dapat mengabaikan data tersebut.

```
df[kolom_kategorical].isnull().sum()
```

```
kategori_pekerjaan    1836
pendidikan               0
status_pernikahan        0
pekerjaan             1843
relationship             0
ras                      0
sex                      0
negara                 583
income                   0
dtype: int64
```

Karena data yang kosong adalah data kategori, maka kita akan menggunakan nilai yang paling banyak muncul sebagai nilai yang missing. Misalnya pada kolom negara, data kita mayoritas berasal dari Indonesia, maka nilai yang kosong diisi dengan Indonesia

```
df['kategori_pekerjaan'].fillna(df['kategori_pekerjaan'].mode()[0], inplace=True)
df['pekerjaan'].fillna(df['pekerjaan'].mode()[0], inplace=True)
df['negara'].fillna(df['negara'].mode()[0], inplace=True)
```

```
df.isnull().sum()
```

```
umur                0
kategori_pekerjaan  0
final_weight        0
pendidikan          0
jumlah_pendidikan   0
status_pernikahan   0
pekerjaan           0
relationship        0
ras                 0
sex                 0
capital_gain        0
capital_loss        0
hours_per_week      0
negara              0
income              0
dtype: int64
```

Setelah itu penulis melakukan feature engineering. Pada algoritma ini, terdapat dua jenis data yaitu kategorikal dan numerik. Algoritma naïve bayes tidak bisa digunakan jika ada data numerik dan gaussian naïve bayes tidak bisa dilakukan jika ada data kategorikal. Jadi kita harus mengubah data kategorikal menjadi numerik.

Penulis akan memisahkan kolom data dan label serta mengubah semua kolom kategorikal pada data menjadi numerik dengan menggunakan metode One Hot Encoding. Sederhananya metode ini akan membuat sejumlah kolom berdasarkan nilai unik pada kolom tersebut. Sebagai contoh kolom jenis kelamin bernilai Laki-Laki dan Perempuan. Maka jika menggunakan One Hot Encoding maka kita akan membuat dua kolom baru yaitu Kelamin_1 dan Kelamin_2. Jika sebuah data bernilai Laki-Laki maka kolom Kelamin_1 akan bernilai 1 dan Kelamin_2 bernilai0. Begitu juga sebaliknya untuk perempuan, kelamin_1 bernilai 0 dan kelamin_2 bernilai 1.

```
from sklearn.preprocessing import LabelBinarizer
X = df.drop(['income'], axis=1)
y = df['income']

kolom_kategorical = ['kategori_pekerjaan', 'pendidikan', 'status_pernikahan', 'pekerjaan', 'relationship', 'ras', 'sex', 'negara']
for var in kolom_kategorical:
  jobs_encoder = LabelBinarizer()
  jobs_encoder.fit(X[var])
  transformed = jobs_encoder.transform(X[var])
  ohe_df = pd.DataFrame(transformed)
  ohe_df.columns = [var+"_"+str(col) for col in ohe_df.columns]
  X = pd.concat([X, ohe_df], axis=1).drop([var], axis=1)
```

Adapun contoh hasil One Hot Encoding pada kolom negara adalah sebagai berikut:

```
ohe_df.head()
```

	negara_0	negara_1	negara_2	negara_3	negara_4	negara_5	negara_6	negara_7	negara_8	negara_9	negara_10	negara_11	negara_12	negara_13	negara_14	negara_15	negara_
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	

Selanjutnya penulis akan melakukan scaling pada data menggunakan RobustScaler. Metode ini menghilangkan median dan menscale data berdasarkan range quantile.

```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()
X = scaler.fit_transform(X)
```

7.4.3 Training dan Evaluasi

Sebelum training, data dipecah menjadi data train dan data test

```
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 = 0.3, random_state = 0)
```

Lalu model dilatih dengan menggunakan data latih (pada baris 5)

```
1  from sklearn.naive_bayes import GaussianNB
2  # instantiate the model
3  gnb = GaussianNB()
4  # fit the model
5  gnb.fit(X_train, y_train)
6
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Setelah itu dilakukan prediksi dan pengukuran akurasi. Akurasi yang didapatkan sebesar 81.07 %

```
1  y_pred = gnb.predict(X_test)
2  y_pred
```

```
array(['<=50K', '<=50K', '>50K', ..., '>50K', '<=50K', '<=50K'],
      dtype='<U5')
```

```
1  from sklearn.metrics import accuracy_score
2  print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score: 0.8107
```

Bab

8

Klasterisasi K-Mean

Computer science is no more about computers than astronomy is about telescopes. – Edsger W. Dijkstra,.

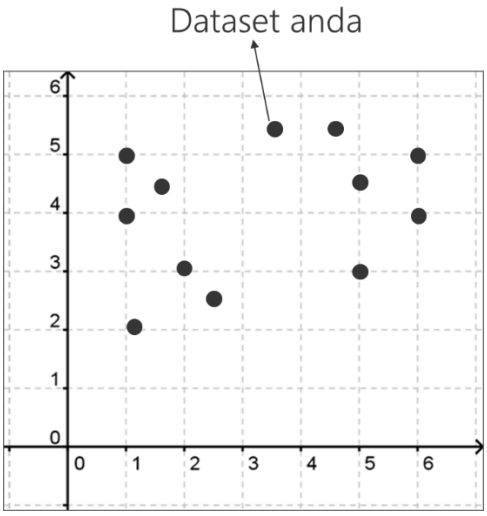
K mean adalah salah satu "*unsupervised machine learning algorithms*" yang paling sederhana dan populer. Tujuan dari algoritma ini adalah untuk menemukan grup dalam data, dengan jumlah grup yang diwakili oleh variabel K. Variabel K sendiri adalah jumlah kluster yang kita inginkan .

Metode K-Means Clustering berusaha mengelompokkan data yang ada ke dalam beberapa kelompok, dimana data dalam satu kelompok mempunyai karakteristik yang sama satu sama lainnya dan mempunyai karakteristik yang berbeda dengan data yang ada di dalam kelompok yang lain. Karakteristik yang sama itu ditandai dengan jarak atau distance yang lebih dekat, mirip seperti KNN.

Dengan kata lain, metode K-Means Clustering bertujuan untuk meminimalisasikan *objective function* yang diset dalam proses clustering dengan cara meminimalkan variasi antar data yang ada di dalam suatu cluster dan memaksimalkan variasi dengan data yang ada di cluster lainnya.

8.1 Algoritma K-Mean

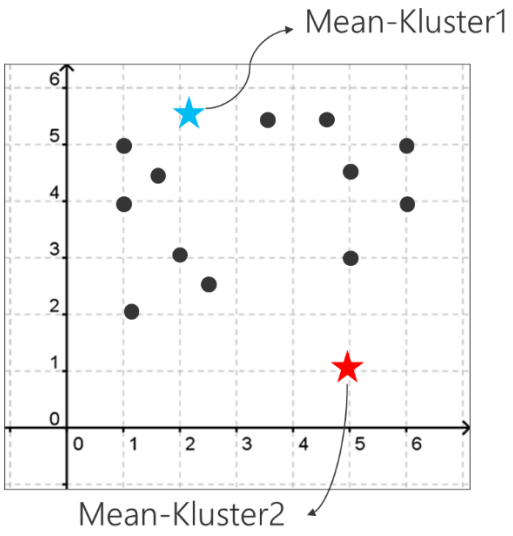
Prosedur yang digunakan dalam melakukan optimasi menggunakan k-means pada data digambar 8.1 adalah sebagai berikut:



Gambar 8.1. Dataset yang akan dikelompokkan

Langkah 1. Tentukan jumlah kluster

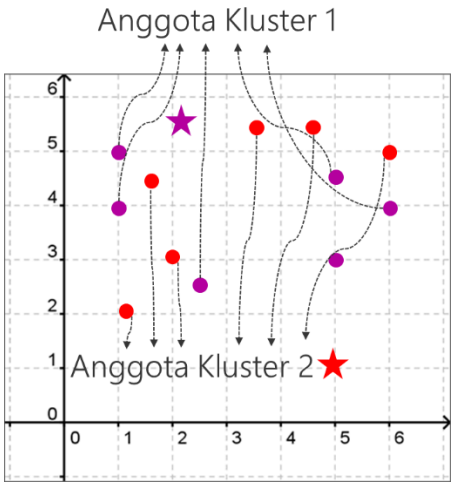
Jumlah kluster adalah jumlah kelompok yang diinginkan. Penentuan jumlah kluster ini merupakan langkah awal yang menentukan hasil dari klasterisasi nantinya. Untuk mengelompokkan data tersebut perlu keahlian dan pemahaman terhadap data dan subjek permasalahan. Namun kita juga dapat melakukan uji terhadap kluster untuk menentukan K-Terbaik. Jumlah kluster yang telah ditentukan akan diberi nilai koordinat acak yang merepresentasikan mean atau rata-rata dari kelompok tersebut. Nilai mean kluster ini kita sebut mean-cluster



Gambar 8.2. Pemilihan mean-kluster secara acak sejumlah K

Langkah 2. Alokasikan data ke dalam kluster secara random

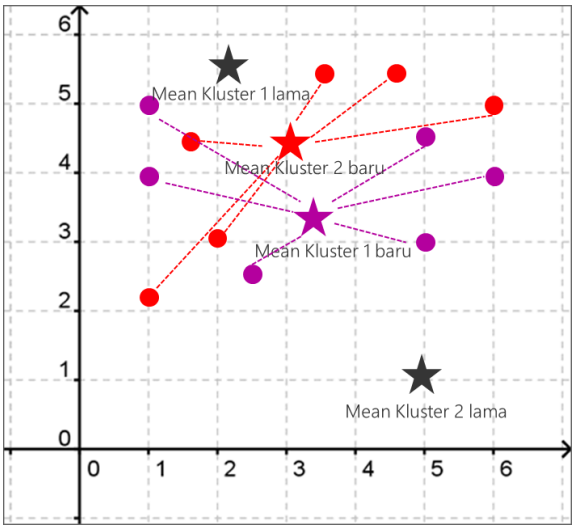
Setiap data dimasukkan ke dalam kluster tertentu secara acak. Walaupun akan nada data yang masuk ke dalam kluster yang salah tidak apa-apa. Karena algoritma ini akan mengubahnya berdasarkan nilai mean-kluster yang di set pada langkah 1 di langkah berikutnya. Ilustrasinya dapat dilihat pada gambar 8.3



Gambar 8.3. Pengalokasian data kedalam kluster secara acak

Langkah 3. Update mean/rata-rata jarak kluster berdasarkan data kluster.

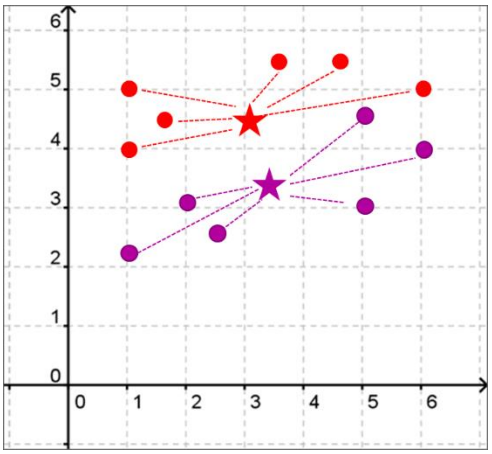
Langkah ini merupakan koreksi untuk langkah 1 dimana kita memberikan nilai acak pada mean-kluster. Proses update dilakukan dengan menghitung jarak rata-rata baru pada kluster tersebut berdasarkan titik-titik pada langkah 2 lalu nilai mean yang baru akan menggantikan nilai mean-cluster. Dengan kata lain, nilai mean-cluster telah di update sesuai dengan anggotanya.



Gambar 8.4. Penentuan mean cluster baru berdasarkan data anggota kluster

Step 4. Alokasikan masing-masing data ke rata-rata terdekat

Langkah ini merupakan koreksi terhadap anggota kluster, dimana keangotaan data terhadap kluster diubah berdasarkan jarak data ke titik mean cluster. Keanggotaan dipilih berdasarkan jarak terdekat. Sebagai contoh pada gambar 10.4, data yang berada pada titik (1,5) adalah anggota kluster 1, namun jika dihitung kembali jaraknya ke mean kluster 1 baru dan mean kluster 2 baru ternyata jaraknya lebih dekat kepada mean kluster 2. Oleh karena itu pada gambar 10.5, keanggotaan titik (1,5) berubah dari kluster 1 menjadi kluster 2



Gambar 8.5. Update keanggotaan kluster berdasarkan mean cluster baru

Step 5. Kembali ke langkah 3

Apabila masih ada data yang berpindah cluster pada langkah 4 atau apabila perubahan nilai centroid, ada yang di atas nilai threshold yang ditentukan atau apabila perubahan nilai pada objective function yang digunakan, di atas nilai threshold yang ditentukan maka Kembali ke langkah 3 untuk mengupdate nilai mean-kluster.

8.2 Implementasi K-Mean

Pada implemetasi ini, penulis akan menggunakan Scikit Learn. Adapun langkah-langkah yang dilakukan adalah:

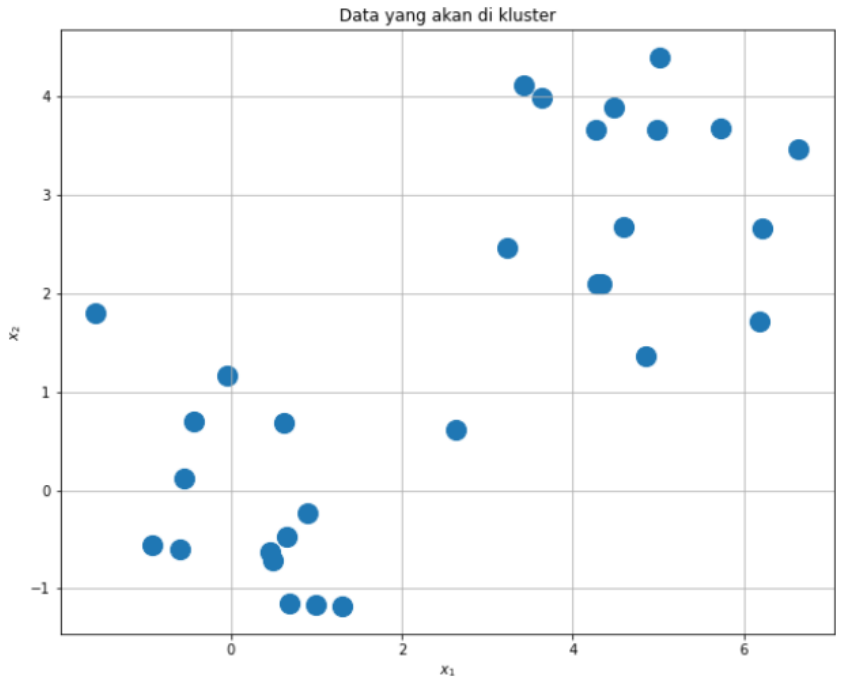
1. Persiapkan dataset

```
[1]  1  import numpy as np
     2  from sklearn.datasets import make_blobs
     3  import matplotlib.pyplot as plt
     4
```

```
[2]  1  #membuat dataset dengan pusat (0,0) dan (5,3)
     2  X,y = make_blobs(n_samples=30, centers=[(0,0),(5,3 )],random_state=86)
```

Pada implementasi ini kita akan membuat data dengan 2 pusat data yaitu (0,0) dan (5,3). Adapun bentuk datanya adalah sebagai berikut

```
[3]  1  plt.figure(figsize=(10, 8))
     2  plt.scatter(X[:, 0], X[:, 1], s = 200,  marker = "o", alpha = 1 )
     3  plt.title("Data yang akan di kluster")
     4  plt.xlabel('$x_1$')
     5  plt.ylabel('$x_2$')
     6  plt.grid(True)
     7
```



2. Klusterisasi

   Kelas KMean membutuhkan hyperparameter n_cluster (jumlah K) yang pada kasus ini ditentukan sebanyak 2. Selanjutnya melakukan pelatihan terhadap data

```
[4]  1  from sklearn.cluster import KMeans
     2  # Inisialisasi model K-Mean dengan K=2
     3  kmeans = KMeans(n_clusters = 2, random_state=123)
     4  # melatih Data
     5  kmeans.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=123, tol=0.0001, verbose=0)
```

3. Menampilkan Hasil Klasifikasi

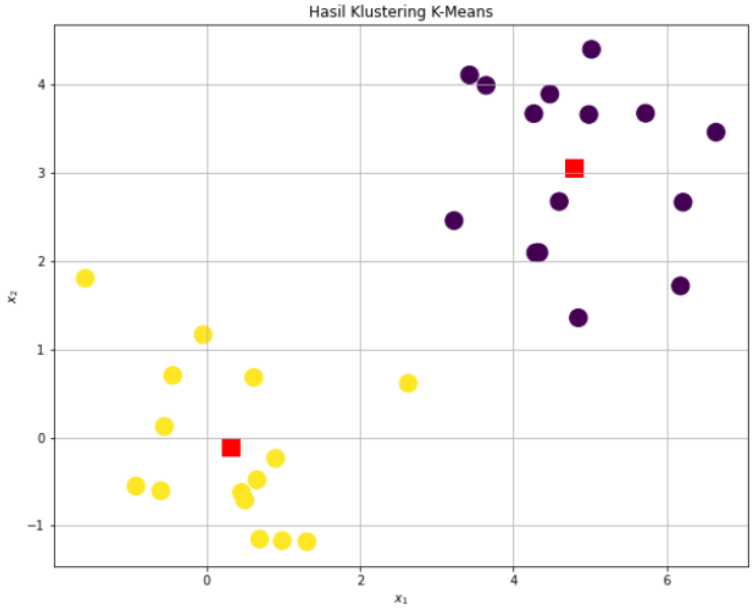
   Hasil pelatihan menunjukkan titik pusat kluster 1 berada di (4.79,3.05) dan kluster 2 berada di (0.308, -0.1082). Agar konvergen, KMEAN membutuhkan 3 iterasi.

```
[5]  1  #Menampilkan pusat cluster
     2  print(kmeans.cluster_centers_)
     3  # Menampilkan Hasil Kluster
     4  print(kmeans.labels_)
     5  #menampilkan jumlah iterasi
     6  print( kmeans.n_iter_)
```

```
[[ 4.79372715  3.05802975]
 [ 0.30821815 -0.10829635]]
[0 1 0 0 0 1 0 1 1 1 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0]
3
```

   Adapun hasil klasifikasi adalah sebagai berikut:

```
[6]  1  plt.figure(figsize=(10, 8))
     2  output = plt.scatter(X[:,0], X[:,1], s = 200, c =  kmeans.labels_, marker = "o", alpha = 1 )
     3  centers = kmeans.cluster_centers_
     4  plt.scatter(centers[:,0], centers[:,1], c='red', s=200, alpha=1 , marker="s");
     5  plt.title("Hasil Klustering K-Means")
     6  plt.xlabel('$x_1$')
     7  plt.ylabel('$x_2$')
     8  plt.grid(True)
     9  plt.show()
```



8.3 Studi Kasus Segmentasi Pelanggan Mall

Penulis akan menggunakan data mengenai pengunjung mall lalu mencoba melakukan klusterisasi dan segmentasi terhadap pelanggan mall tersebut. Langkah pertama adalah membaca dataset dan melakukan data preprocesing. Lalu yang terakhir proses segmentasi terhadap pelanggan.

```
1  import pandas as pd
2  import seaborn as sns
3  import matplotlib.pyplot as plt
4  from sklearn.preprocessing import StandardScaler, LabelEncoder
5  df = pd.read_csv('Mall_Customers.csv')
6  df.head(10)
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

8.3.1 Data Preprocessing

Dataset ini terdiri atas 5 kolom, dimana kolom pertama berupa CustomerID sebagai penanda pelanggan, Gender yaitu jenis kelamin pelanggan, Age yaitu umur pelanggan, Annual Income (k$) yaitu pendapatan pertahun, dan Spending Score (1–100) yaitu nilai yang diberikan oleh mall kepada pelanggan sebagai penilaian seberapa besar seseorang mengeluarkan uang di mall.

Pertama-tama, kita harus mengecek apakah data tersebut ada missing value atau duplicated value

```
[2]  1  #Missing values
     2  print("missing value :")
     3  print(df.isna().sum())
     4  #Cek Duplikasi Data
     5  print("jumlah duplikasi = ",df.duplicated().sum())
```

```
missing value :
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
jumlah duplikasi =  0
```

```
[3]  1  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

Customer ID berisi ID perlanggan dan tidak ada relasi dengan kelompok pelanggan. Oleh karena itu kita tidak membutuhkan kolom CustomerID, maka penulis akan menghilangkan kolom tersebut. Selain itu, untuk mempermudah pekerjaan nantinya saya akan merubah nama kolom *Annual Income* (k$) dan Spending Score (1–100) menjadi *Annual Income* dan *Spending Score* saja.

Selanjutnya, pada kolom *Gender* terdapat dua nilai yakni *Male* dan *Female*, maka penulis menggunakan LabelEncoder untuk merubah Female menjadi value 0 dan Male menjadi value 1. Perubahan ini dilakukan karena Algoritma K-Means model membutuhkan nilai numerik untuk dapat bekerja. LabelEncoder sendiri bekerja dengan cara merubah seluruh kategori

yang ada di dalam suatu kolom menjadi angka. Angka tersebut diurutkan berdasarkan huruf abjad dari kategori yang ada.

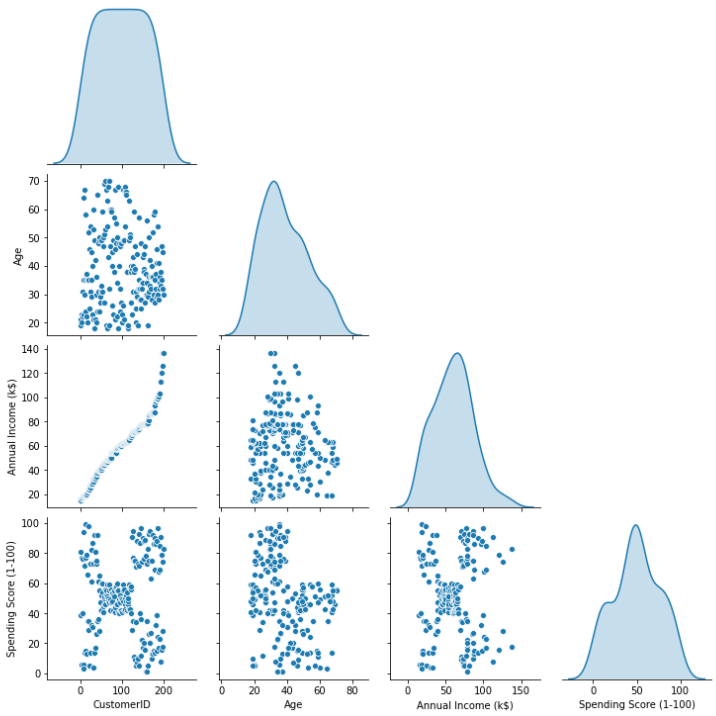
```
[4]  #menghapus feature CustomerID
     df.drop('CustomerID', axis = 1, inplace = True)
     #rename kolom
     df.rename(columns = {'Annual Income (k$)': 'Annual Income'}, inplace = True)
     df.rename(columns = {'Spending Score (1-100)': 'Spending Score'}, inplace = True)

     #merubah nilai male dan female menjadi numeric
     from sklearn.preprocessing import StandardScaler, LabelEncoder
     labeler = LabelEncoder()
     df['Gender'] = labeler.fit_transform(df['Gender'])
     df
```

	Gender	Age	Annual Income	Spending Score
0	1	19	15	39
1	1	21	15	81
2	0	20	16	6
3	0	23	16	77
4	0	31	17	40
...	...	...	...	...
195	0	35	120	79
196	0	45	126	28
197	1	32	126	74
198	1	32	137	18
199	1	30	137	83

200 rows × 4 columns

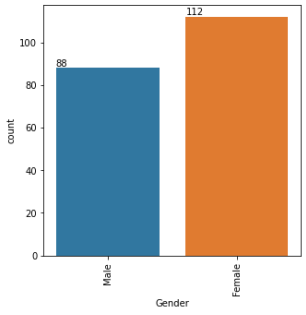
Selanjutnya kita akan melihat sebaran data untuk memastikan data telah siap untuk dilakukan pelatihan

```
sns.pairplot(df,corner=True,diag_kind="kde")
plt.show()
```

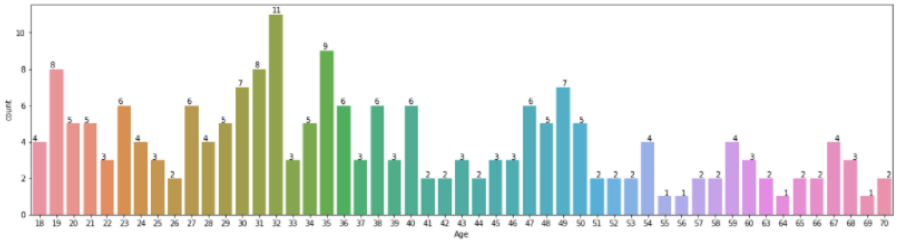


Hal menarik yang dapat saya lihat dari data dan visualisasi data diatas adalah:

1. Pengunjung Mall lebih banyak perempuan/*female* berjumlah 112 sedangkan laki-laki berjumlah



2. Rentang umur pengunjung dari 18 sampai 70 tahun, namun sebagian besar pelanggan Mall dengan rentang umur yang lebih muda



3. Sebagian besar pelanggan mall memiliki pendapatan dibawah 100K$
4. Terdapat Normal distribusi untuk Spending Score pelanggan, yaitu mendekati nilai 50
5. Hubungan antara kolom *Annual Income* dan *Spending Score* membentuk suatu kluster yang terlihat secara langsung membentuk 5 kelompok.

Selanjutnya kita lihat statistic dari masing-masing kolom

```
1   df.describe()
```

	Gender	Age	Annual Income	Spending Score
count	200.000000	200.000000	200.000000	200.000000
mean	0.440000	38.850000	60.560000	50.200000
std	0.497633	13.969007	26.264721	25.823522
min	0.000000	18.000000	15.000000	1.000000
25%	0.000000	28.750000	41.500000	34.750000
50%	0.000000	36.000000	61.500000	50.000000
75%	1.000000	49.000000	78.000000	73.000000
max	1.000000	70.000000	137.000000	99.000000

Karena jarak data beragam, maka kita akan melakukan normalisasi terhadap data menggunakan Z-score sehingga data memiliki mean 0 dan standard deviation 1. Semua kolom akan di normalisasi kecuali Gender.

```
#Scaling Data
scaler = StandardScaler()
# scaling kecuali gender
Gender = df['Gender']
df.drop('Gender', axis = 1)
#proses scaling
df = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)
df['Gender'] = Gender
df.head(5)
```

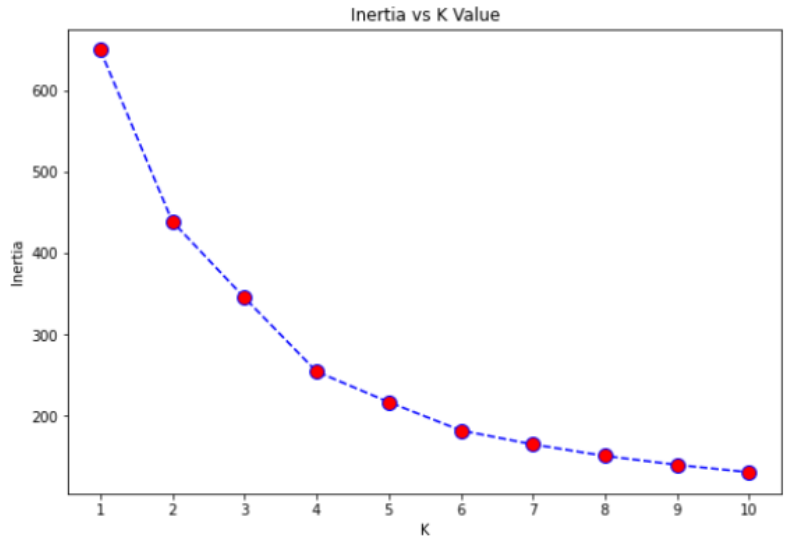
	Gender	Age	Annual Income	Spending Score
0	1	-1.424569	-1.738999	-0.434801
1	1	-1.281035	-1.738999	1.195704
2	0	-1.352802	-1.700830	-1.715913
3	0	-1.137502	-1.700830	1.040418
4	0	-0.563369	-1.662660	-0.395980

8.3.2 Training Model

Sebelum memulai pelatihan berarti kita harus menentukan jumlah kluster. Namun, pada kasus ini kita tidak mengetahuinya oleh karena itu kita dapat menggunakan elbow method.

```
from sklearn.cluster import KMeans
inertia_list = []
for i in range(1,11):
  kmeans = KMeans(n_clusters = i)
  kmeans.fit(df)
  inertia_list.append(kmeans.inertia_)

plt.figure(figsize = (9,6))
plt.plot(range(1,11), inertia_list, color = 'blue',
         linestyle = 'dashed', marker ='o',
         markerfacecolor = 'red', markersize = 10)
plt.title('Inertia vs K Value')
plt.xticks(range(1,11, 1))
plt.xlabel('K')
plt.ylabel('Inertia')

```

```
Text(0, 0.5, 'Inertia')
```



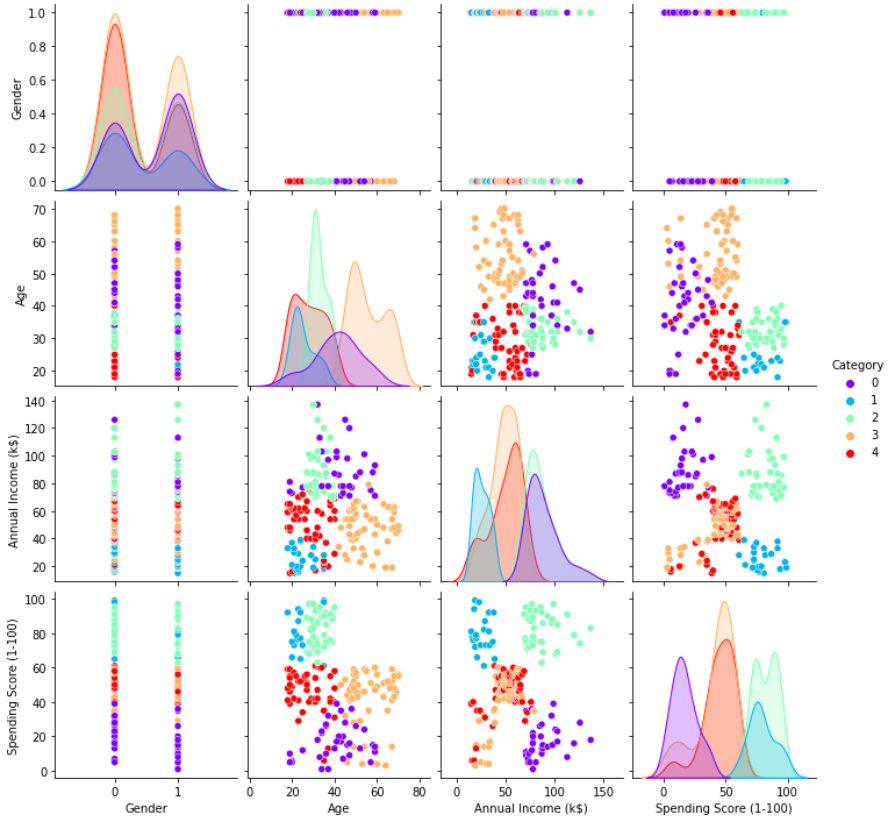
Pemilihan jumlah kluster yang sesuai, kita bisa menggunakan evaluasi apa yang kita sebut Inertia. Inertia adalah total jarak dari suatu *centroid* kepada data yang berada di kluster yang sama. Kita ingin Inertia yang lebih kecil karena kita mengingkan kluster dengan data-data yang berdekatan.

Elbow Method mempunyai nama seperti itu karena plot yang dihasilkan memiliki bentuk seperti siku tangan, dimana semakin meningkat nilai K kita, Inertia juga semakin mengecil sampai ke titik tertentu.

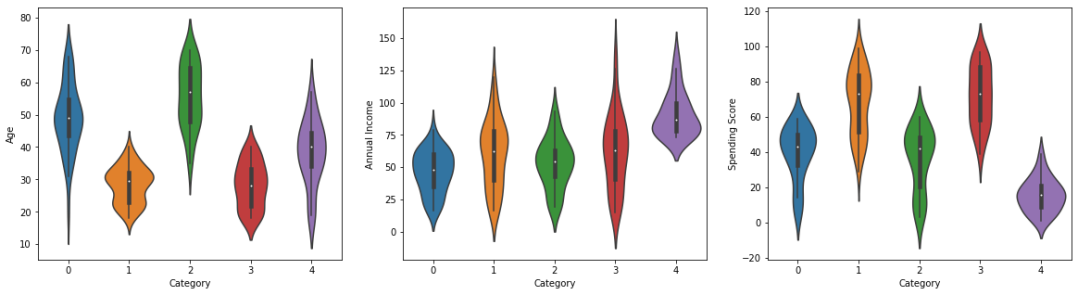
Tentu, ada pertanyaan lebih lanjut lagi jika kita menggunakan *Elbow Method* ; sampai seberapa besarkah jumlah K yang baik jika dengan penambahan nilai K, Inerita akan semakin turun? **Pertama** , kita secara pribadi dapat menentukkan batas nilai K yang seharusnya. Kita dapat terus menurunkan nilai K, tetapi tentu saja kita bisa membatasi nilai K sesuai pengetahuan dan logika kita. **Kedua** , nilai K dapat dipilih pada saat K di nilai yang sangat menurunkan Inertia; sebagai contoh, jika gambar di atas kita dapat memilih antara K di nilai 4–6 karena di antara nilai itu Inertia mengalami penurunan yang signifikan dan setelah nilai 6, penurunan Inertia yang terjadi tidak terlalu kecil.

Sekarang kita akan mencoba untuk mengelompokkan data kita dan mengevaluasi kluster yang dihasilkan dengan cara kluster tersebut dapat dikembalikan ke data awal.

```
km = KMeans(n_clusters =5).fit(df)
#Membaca kembali original data ke variable lain
#karena data untuk klustering telah diubah2
df_ori = pd.read_csv('Mall_Customers.csv')
df_ori.drop('CustomerID', axis = 1, inplace = True)
df_ori.rename(columns = {'Annual Income (k$)': 'Annual Income','Spending Score (1-100)': 'Spending Score'}, inplace = True)

#Membuat satu kolom category untuk menampung hasil kluster kita
df_ori['Category'] = km.labels_
sns.pairplot(data = df_ori, hue = 'Category', palette = 'rainbow')
```




```
fig, axes = plt.subplots(1,3, figsize=(20,5))

sns.violinplot(x = 'Category', y = 'Age', data = df_ori,ax=axes[0])
sns.violinplot(x = 'Category', y = 'Annual Income', data = df_ori,ax=axes[1])
sns.violinplot(x = 'Category', y = 'Spending Score', data=df_ori,ax=axes[2])
plt.show()
```



```
df_ori[['Age', 'Annual Income','Spending Score','Category']].groupby('Category').mean()
```

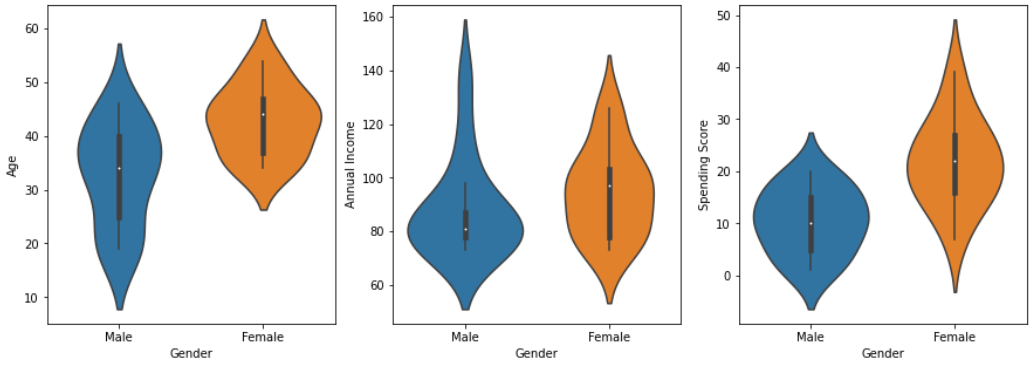
Category	Age	Annual Income	Spending Score
0	37.769231	91.461538	16.115385
1	55.628571	52.828571	36.200000
2	28.392857	60.428571	68.178571
3	28.250000	62.000000	71.675000
4	49.325581	47.000000	38.813953

Dengan melihat plot diatas, kita dapat memberikan kesimpulan dari pembagian data kita sebagai berikut:

1. Kelompok 1 (Category=0)

   Memiliki rata-rata umur 37 tahun, dengan rata-rata *annual income* $91 K dan *spending score* 40.

```
group_1= df_ori[df_ori['Category']==0]
group_1.head()

fig, axes = plt.subplots(1,3, figsize=(15,5))

sns.violinplot(x = 'Gender', y = 'Age', data = group_1,ax=axes[0])
sns.violinplot(x = 'Gender', y = 'Annual Income', data = group_1,ax=axes[1])
sns.violinplot(x = 'Gender', y = 'Spending Score', data=group_1,ax=axes[2])
plt.show()
```

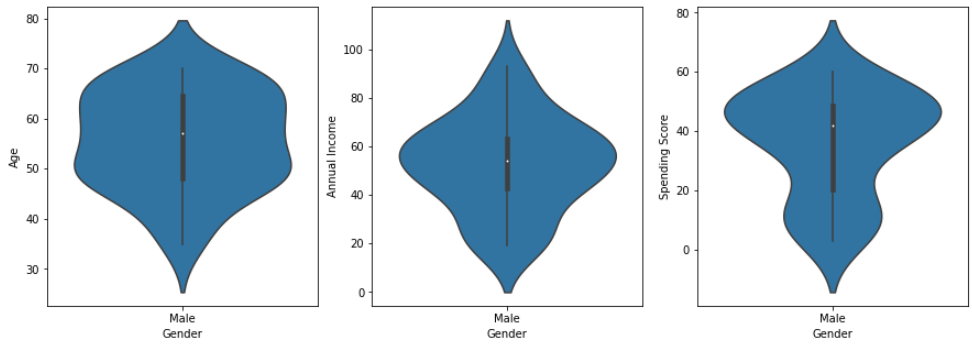


Dapat dikatakan kelompok ini dikatakan memiliki kelompok pria dan wanita paruh baya yang memiliki annual income yang tinggi dan spending yang rendah.

2. Kelompok 2 (Category=1)

   Kelompok ini memiliki rata-rata umur 55 tahun dengan kelamin perempuan. Rata-rata annual income dan spending score adalah $52K dan 36.

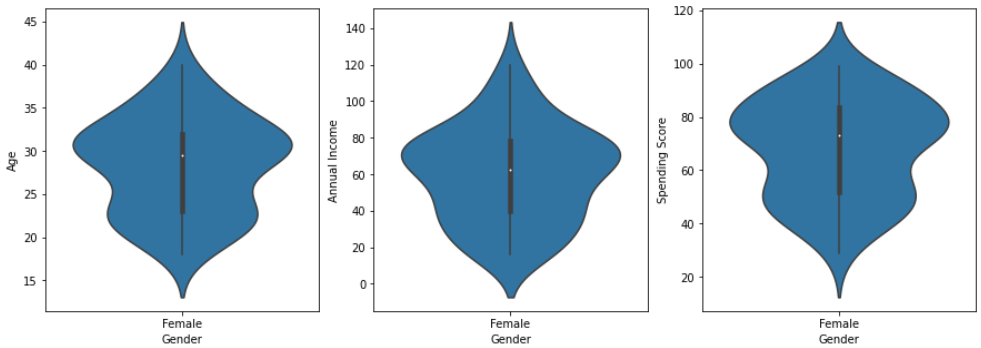
```
group_2= df_ori[df_ori['Category']==1]

fig, axes = plt.subplots(1,3, figsize=(15,5))

sns.violinplot(x = 'Gender', y = 'Age', data = group_2,ax=axes[0])
sns.violinplot(x = 'Gender', y = 'Annual Income', data = group_2,ax=axes[1])
sns.violinplot(x = 'Gender', y = 'Spending Score', data=group_2,ax=axes[2])
plt.show()
```



   Kelompok ini adalah kelompok pria paruh baya yang memiliki pendapatan medium dengan spending medium

3. Kelompok 3 (Category=2)

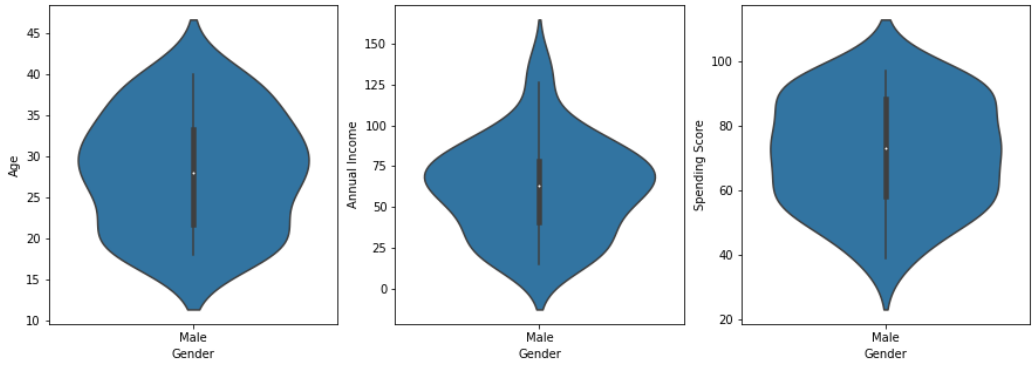
```
group_3= df_ori[df_ori['Category']==2]

fig, axes = plt.subplots(1,3, figsize=(15,5))

sns.violinplot(x = 'Gender', y = 'Age', data = group_3,ax=axes[0])
sns.violinplot(x = 'Gender', y = 'Annual Income', data = group_3,ax=axes[1])
sns.violinplot(x = 'Gender', y = 'Spending Score', data=group_3,ax=axes[2])
plt.show()
```



Kelompok ini adalah kelompok wanita dengan pendapatan menengah keatas namun memiliki spending tinggi

4. Kelompok 4 (category=3)

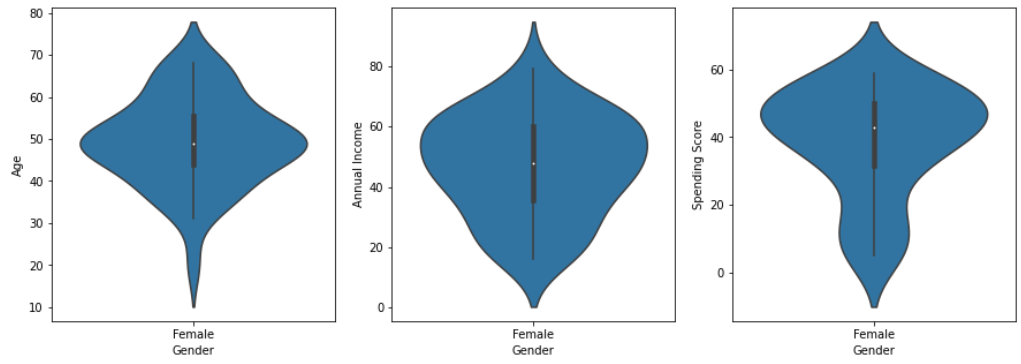
```
group_4= df_ori[df_ori['Category']==3]

fig, axes = plt.subplots(1,3, figsize=(15,5))

sns.violinplot(x = 'Gender', y = 'Age', data = group_4,ax=axes[0])
sns.violinplot(x = 'Gender', y = 'Annual Income', data = group_4,ax=axes[1])
sns.violinplot(x = 'Gender', y = 'Spending Score', data=group_4,ax=axes[2])
plt.show()
```



Kelompok ini adalah kelompok pria dengan pendapatan menengah keatas namun memiliki spending tinggi

5. Kelompok 4 (category=3)

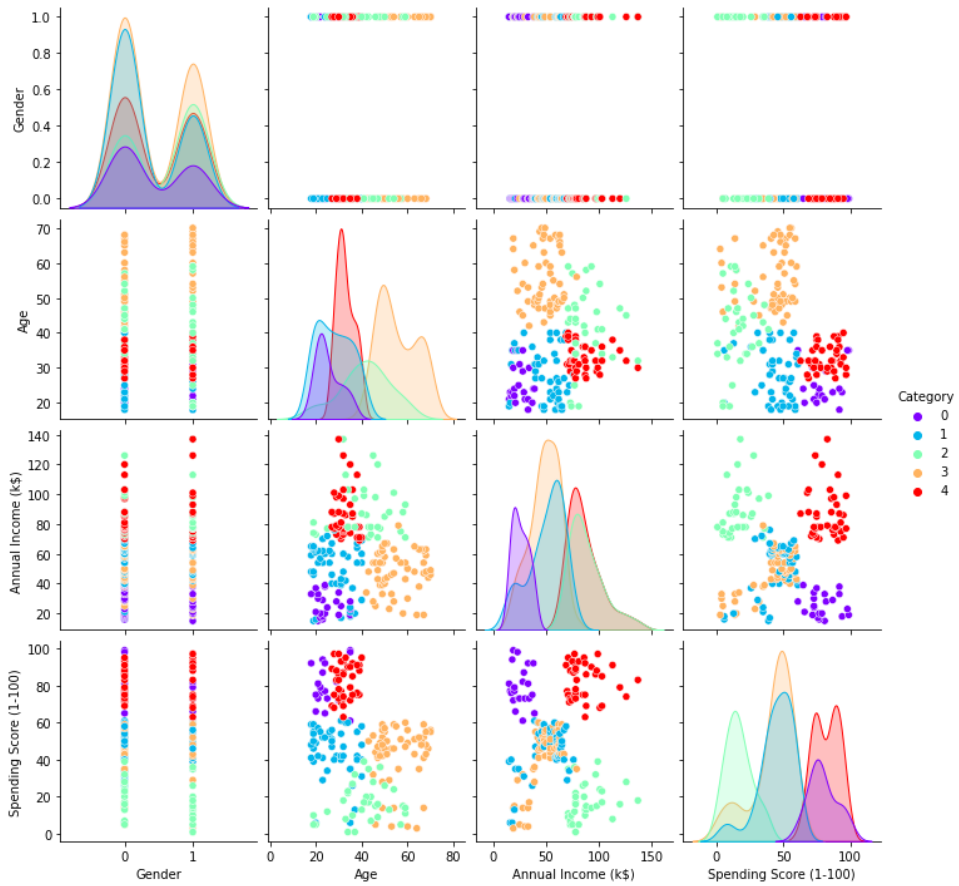
```
group_5= df_ori[df_ori['Category']==4]

fig, axes = plt.subplots(1,3, figsize=(15,5))

sns.violinplot(x = 'Gender', y = 'Age', data = group_5,ax=axes[0])
sns.violinplot(x = 'Gender', y = 'Annual Income', data = group_5,ax=axes[1])
sns.violinplot(x = 'Gender', y = 'Spending Score', data=group_5,ax=axes[2])
plt.show()
```



Kelompok ini adalah kelompok wanita dengan pendapatan menengah kebawah namun memiliki spending rendah

Tidak ada syarat yang mutlak untuk menggunakan seluruh data dalam membuat kluster. Jika dilihat dari data kita diatas, akan mudah memang untuk K-Means membagi data dengan menggunakan kolom gender karena kolom tersebut hanya memiliki 2 nilai (0 (Female) dan 1 (Male)).

```
df.drop('Gender', axis = 1)
km = KMeans(n_clusters =5).fit(df)
#Membaca kembali original data ke variable lain
mall_ori = pd.read_csv('Mall_Customers.csv')
mall_ori.drop('CustomerID', axis = 1, inplace = True)
mall_ori['Gender'] = labeler.fit_transform(mall_ori['Gender'])
#Membuat satu kolom category untuk menampung hasil kluster kita
mall_ori['Category'] = km.labels_
sns.pairplot(data = mall_ori, hue = 'Category', palette = 'rainbow')
```



Sekarang, kita dapat mengambil kesimpulan dari hasil kluster diatas tanpa gender sebagai berikut:

1. Kelompok 0 merepresentasikan pelanggan dengan *annual income* sangat rendah, *spending score* rendah, dan rentang umur yang beragam
2. Kelompok 1 merepresentasikan pelanggan dengan *annual income* agak rendah, *spending score* agak rendah, dan memiliki rentang umur menengah hinnga tua
3. Kelompok 2 merepresentasikan pelanggan dengan rentang umur menengah ke arah muda, *annual income* menengah hingga tinggi, *serta spending score* yang tinggi
4. Kelompok 3 merepresentasikan pelanggan dengan rentang umur muda, *annual income* yang rendah, serta *spending score* yang menengah hingga tinggi
5. Kelompok 4 merepresentasikan pelanggan dengan umur yang beragam, *annual income* menengah hingga tinggi, serta *spending score* yang rendah

Dengan merubah data yang dilatih, kita mampu mendapatkan perspektif baru. Selain itu, dengan merubah nilai K juga, kita dapat melihat kelompok baru yang belum dapat kita lihat

Daftar Pustaka

[1] Kevin P. Murphy, Machine Learning: A Probabilistic Perspective, 2012.

[2] W. Richert, L.P. Coelho, Building Machine Learning Systems with Python.

[3] J. Grus, Data Science from Scratch, O'Reilly Media, United States of America, 2015.

[4] P. Xanthopoulos, P.M. Pardalos, T.B. Trafalis, Robust Data Mining, Springer New York, New York, NY, 2013, doi:10.1007/978-1-4419-9878-1.

[5] The Art of Feature Engineering: Essentials for Machine Learning.

[6] C.C. Aggarwal, Mining Text Data, Springer US, Boston, MA, 2012, doi:10.1007/978-1-4614-3223-4.

[7] E. Duchesnay, T. Löfstedt, "Statistics and Machine Learning in Python," 333, 2018.

[8] B. Ratner, S. Day, C. Davies, Statistical and Machine-Learning Data Mining, 2011, doi:10.1201/b11508.

[9] D.G. PASCUAL, Artificial intelligence tools, 2015, doi:10.1007/978-3-642-28085-6_3.

[10] H. Sahli, An Introduction to Machine Learning, 2020, doi:10.1002/9781119720492.ch7.

[11] Christopher M. Bishop F.R.Eng, Pattern Recognition and Machine Learning, 2006, doi:10.13109/9783666604409.185.

Indek