3

Data Analytics

> "Hiding within those mounds of data is knowledge that could change the life of a patient, or change the world"
>
> Atul Butte

Bab ini memuat penjelasan tahapan-tahapan umum untuk analisis data, serta beberapa karakteristik tipe data. Materi pada bab ini dapat dianggap sebagai kerangka berpikir (*framework*) atau langkah kerja. Karena buku ini hanya bersifat pengantar, materi yang disampaikan mungkin kurang lengkap. Penulis menyarankan pembaca untuk membaca buku oleh Witten et al. [18] dan Jeff Leek [19].

3.1 Pengenalan Data Analytics

Secara umum, subbab ini adalah ringkasan dari buku Jeff Leek [19]. Untuk detailnya, kamu dapat membaca buku tersebut secara langsung. Penulis merekomendasikan buku tersebut karena ringkas dan mudah dipahami bahkan oleh pemula.

Kita tahu di dunia ini ada banyak masalah. Masalah adalah ketika tujuan yang diinginkan tidak tercapai (*current state* bukanlah *desired state*). Agar *current state* menjadi *desired state*, kita melakukan kegiatan yang disebut penyelesaian masalah (*problem solving*). Tiap bidang (domain) mendefinisikan permasalahan secara berbeda. Oleh karena itu, mengetahui teknik *machine learning* tanpa mengetahui domain aplikasi adalah sesuatu yang kurang baik (semacam buta). Kamu memiliki ilmu, tetapi tidak tahu ilmunya mau digunakan untuk apa. Contohnya, bidang keilmuan pemrosesan bahasa alami (*natural language processing*) menggunakan *machine learning* untuk mengklasifikasikan teks; bidang keilmuan pemrosesan suara menggunakan *machine*

learning untuk mentranskripsikan suara manusia menjadi teks. Tiap bidang merepresentasikan permasalahan ke dalam formulasi yang berbeda. Bisa jadi bentuk komputasi (representasi) pada bidang satu berbeda dengan bidang lainnya. Hal ini perlu kamu ingat karena interpretasi representasi sangat bergantung pada konteks permasalahan (domain). Buku ini adalah pengenalan teknik yang bersifat umum.

Seperti yang sudah dijelaskan pada bab-bab sebelumnya. *Machine learning* adalah inferensi berdasarkan data. *Raw data* atau data mentah adalah sekumpulan fakta (*record*, *event*) yang kemungkinan besar tidak memberikan penjelasan apapun. Sama halnya dengan kebanyakan data di dunia nyata, *raw data* bersifat tidak rapih, misalnya mengandung *missing value* atau ada data yang tidak memiliki label padahal data lainnya memiliki label (ingat kembali materi bab 1). Agar mampu menganalisis *raw data* menggunakan teknik *machine learning*, pertama-tama kita harus merapikan data sesuai dengan format yang kita inginkan (*dataset*). Setelah itu, kita menggunakan teknik-teknik yang ada untuk menemukan pola-pola yang ada di data. Dalam komunitas peneliti basis data, dipercaya bahwa data memiliki sangat banyak relasi yang mungkin tidak bisa dihitung. Teknik *machine learning* hanya mampu mengeksplorasi sebagian relasi yang banyak itu. Lalu, kita analisis informasi yang kita dapatkan menjadi pengetahuan yang digunakan untuk memecahkan permasalahan atau membuat keputusan.

Setelah kita menganalisis data dan mendapatkan pengetahuan baru, pengetahuan yang kita temukan dari data pada umumnya dipresentasikan (konferensi, rapat, dsb). Hal umum yang dipaparkan saat presentasi, yaitu:

1. *Performance measure*. Seberapa "bagus" model/ metode yang kamu buat/ ajukan, dibandingkan menggunakan teknik-teknik yang sudah ada. *Performance measure* biasa disajikan dalam bentuk tabel. Perhatikan, mengatakan model/metode kamu "lebih bagus" adalah suatu hal subjektif, untuk itu gunakanlah metode kuantitatif, seperti *p-value* dalam statistik (*hypothesis testing*[1]) untuk mengatakan bahwa memang metode kamu lebih baik (berbeda) dari *baseline*.

2. Tren. Bagaimana pola-pola umum yang ada di data, sesuai dengan tujuan analisis (permasalahan). Biasa disajikan dalam bentuk teks, kurva, atau grafik.

3. *Outlier*. Sajikan data-data yang "jarang" atau tidak sesuai dengan tren yang ada. Apa beda sifat data *outlier* ini dibanding data pada tren? Kamu harus menganalisis hal ini untuk meningkatkan *performance measure* pada penelitian/analisis mendatang. Apakah *outlier* ini penting untuk diurus atau bisa dipandang sebelah mata tanpa membahayakan keputusan/sistem? Tidak jarang kamu mendapat inspirasi untuk meningkatkan

[1] `https://onlinecourses.science.psu.edu/statprogram/node/138`

id	outlook	temperature	humidity	windy	play (class)
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

Tabel 3.1. Contoh dataset *play tennis* (UCI machine learning repository)

kinerja sistem setelah menganalisis *outlier*.

Langkah kerja yang dijelaskan ini adalah pekerjaan rutin *data scientist*. Penulis ingin menekankan sekali lagi, bahwa memahami *machine learning* saja tidak cukup, **kamu harus memahami domain permasalahan** agar mampu melakukan analisis dengan tepat. Terdapat banyak hal yang hanya mampu kamu pahami dari menganalisis data, apabila kamu mengerti domain aplikasi.

3.2 Nilai Atribut dan Transformasi

Perhatikan Tabel 3.1 yang merupakan contoh *dataset* pada *machine learning*. **Dataset** adalah kumpulan data. Seorang anak ingin bermain tenis, tetapi keputusannya untuk bermain tenis (*play*) tergantung pada empat variabel {*outlook, temperature, humidity, windy*}. Keempat variabel ini disebut **fitur**. Setiap fitur memiliki **atribut** nilai dengan **tipe data** dan **range** tertentu. Keputusan untuk bermain (*play*) disebut sebagai label atau kelas (*class*). Pada bab 1 kamu telah diperkenalkan *supervised learning* dan *unsupervised learning*. Pada *supervised learning*, kita ingin mengklasifikasikan apakah seorang anak akan bermain atau tidak, diberikan fitur-fitur yang memuat kondisi observasi. Pada *unsupervised learning*, informasi kolom *play* tidak diberikan, kita harus mengelompokkan data tersebut sesuai dengan fitur-fiturnya (contoh lebih nyata diberikan pada bab 4).

Dari segi data statistik, terdapat beberapa tipe atribut [20]:

1. **Nominal**. Nilai atribut bertipe nominal tersusun atas simbol-simbol yang berbeda, yaitu suatu himpunan terbatas. Sebagai contoh, fitur *outlook*

pada Tabel 3.1 memiliki tipe data **nominal** yaitu nilainya tersusun oleh himpunan {*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. **Ordinal**. Nilai ordinal memiliki urutan, sebagai contoh $4 > 2 > 1$. Tetapi jarak antar suatu tipe dan nilai lainnya tidak harus selalu sama, seperti $4-2 \neq 2-1$. Atribut ordinal kadang disebut sebagai **numerik** atau **kontinu**.

3. **Interval**. Tipe interval memiliki urutan dan *range* nilai yang sama. Sebagai contoh $1-5, 6-10, dst$. 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 $\{1-20, 21-40, \ldots, 81-100\}$. Setiap interval kita beri nama, misal interval $81-100$ diberi nama *nilai A*, interval $61-80$ diberi nama *nilai B*.

4. **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.

Secara umum, data pada *machine learning* adalah nominal atau numerik (ordinal). Variabel yang kita prediksi yaitu *play* disebut **kelas/*class*/label**. Untuk setiap baris pada Tabel 3.1, baris kumpulan nilai variabel non-kelas disebut **vektor fitur/*feature vector***. Contohnya pada Tabel 3.1 $id = 4$, *feature vector*-nya adalah {*outlook=rainy, temperature=mild, humidity=high, windy=false*}. *Feature vector* adalah representasi dari suatu observasi/data. Pada *machine learning*, kita melakukan operasi terhadap data pada representasi *feature vector*-nya. Kami serahkan pada pembaca untuk mencari contoh dataset dengan tipe numerik sebagai pekerjaan rumah[2].

3.3 Ruang Konsep

Dengan data yang diberikan, kita ingin melakukan generalisasi aturan/ konsep yang sesuai dengan data. Hal ini disebut sebagai *inductive learning*. Cara paling sederhana untuk *inductive learning* adalah mengenumerasi seluruh kemungkinan kombinasi nilai sebagai *rule*, kemudian mengeleminasi *rule* yang tidak cocok dengan contoh. Metode ini disebut *list-then-eleminate*. Silahkan

[2] `https://www.cs.waikato.ac.nz/ml/weka/datasets.html`

baca buku Tom Mitchell [4] untuk penjelasakn lebih rinci. Kemungkinan kombinasi nilai ini disebut sebagai ruang konsep (***concept space***). Sebagai contoh pada Tabel 3.1 himpunan nilai masing-masing atribut yaitu:

- $outlook = \{sunny, overcast, rainy\}$
- $temperature = \{hot, mild, cold\}$
- $humidity = \{high, normal\}$
- $windy = \{true, false\}$
- $play = \{yes, no\}$

sehingga terdapat $3 \times 3 \times 2 \times 2 \times 2 = 72$ kemungkinan kombinasi. Tentunya kita tidak mungkin mengenumerasi seluruh kemungkinan kombinasi nilai karena secara praktikal, atribut yang digunakan banyak. Terlebih lagi, apabila mengenumerasi kombinasi atribut bertipe numerik.

Ada algoritma lain yang mendaftar "seluruh kemungkinan kombinasi" bernama *candidate-elemination algorithm* yang lebih efisien dibanding *list-then-eliminate*. Akan tetapi, algoritma ini *computationally expensive* secara praktikal, dalam artian memiliki kompleksitas yang besar dan tidak bisa menyelesaikan permasalahan nyata. Kamu dapat membaca algoritma ini pada buku Tom Mitchell [4] juga. Selain *inductive learning*, kita juga dapat melakukan *deductive learning* yaitu melakukan inferensi dari hal general menjadi lebih spesifik. Walau demikian, secara praktis kita sebenarnya melakukan *inductive learning*.

3.4 Linear Separability

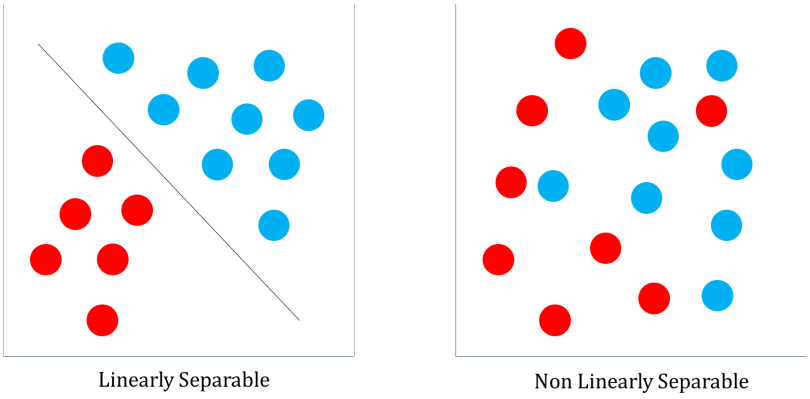
`id`	`humidity`	`windy`	`swim (class)`
1	high	high	yes
2	normal	normal	no

Tabel 3.2. Contoh dataset *linearly separable*

Perhatikan Tabel 3.2. Data pada tabel tersebut kita sebut ***linearly separable***. Sederhananya, untuk suatu nilai tertentu, fitur hanya berkorespondensi dengan kelas tertentu. Ambil contoh pada Tabel 3.2, saat *humidity=high* maka *swim=yes*. Secara "geometris", bila kita proyeksikan *feature vector* ke suatu ruang dimensi, memisahkan kelas satu dan kelas lainnya dapat diperoleh dengan cara menciptakan garis linier (*linear line* – secara lebih umum, menggunakan *hyperplane*[3]). Ilustrasi dapat dilihat pada Gambar 3.1. Sementara pada Tabel 3.1, bila kita hanya melihat fitur *humidity* saja, ketika *humidity=high* bisa jadi *play=yes* atau *play=no*. Kasus ini disebut ***non-linearly separable***.

[3] `https://en.wikipedia.org/wiki/Hyperplane`

Hidup kita tentu akan mudah apabila seluruh data bersifat *linearly separable*, sayangnya kebanyakan data yang ada bersifat *non-linearly separable*.



Gambar 3.1. *Linearly* vs *Non-Linearly Separable*

Untuk memudahkan proses pada data *non-linearly separable*, kita pertamatama mentransformasikan data menjadi *linearly-separable*. Kita dapat menggunakan teknik transformasi data menggunakan *kernel function* seperti *radial basis function*[4]. Pada umumnya, *kernel function* mentransformasi data menjadi lebih tinggi (semacam menambah fitur). Misal dari data yang memiliki dua fitur, ditransformasi menjadi memiliki tiga fitur. Akan tetapi, hal ini tidak praktikal untuk banyak kasus (dijelaskan pada bab 11). Cara lainnya adalah memisahkan data menggunakan model non-linear, contoh: *artificial neural network*. Hal ini penting dipahami karena data yang bersifat *linearly separable* mudah dipisahkan satu sama lain sehingga mudah untuk melakukan *classification* atau *clustering*.

3.5 Seleksi Fitur

Pada subbab sebelumnya, telah dijelaskan bahwa kita dapat mentransformasi data *non-linearly separable* menjadi *linearly separable* dengan cara menambah dimensi data. Pada bab ini, penulis akan menjelaskan justru kebalikannya! Pada permasalahan praktis, kita seringkali menggunakan banyak fitur (*computationally expensive*). Kita ingin menyederhanakan fitur-fitur yang digunakan, misalkan dengan memilih subset fitur awal, atas dasar beberapa alasan [19, 4][5]:

1. Menyederhanakan data/model agar lebih mudah dianalisis.

[4] Silakan baca teknik transformasi lebih lanjut pada literatur lain.
[5] `https://en.wikipedia.org/wiki/Feature_selection`

2. Mengurangi waktu *training* (mengurangi kompleksitas).
3. Menghindari *curse of dimensionality*. Hal ini dijelaskan lebih lanjut pada bab 12.
4. Menghapus fitur yang tidak informatif.
5. Meningkatkan generalisasi dengan mengurangi *overfitting*.

Salah satu contoh cara seleksi fitur adalah menghapus atribut yang memiliki *variance* bernilai 0. Berdasarkan *information theory* atau *entropy*, fitur ini tidak memiliki nilai informasi yang tinggi. Dengan kata lain, atribut yang tidak dapat membedakan satu kelas dan lain bersifat tidak informatif. Kamu dapat membaca beberapa contoh algoritma seleksi fitur pada library sklearn[6].

3.6 Classification, Association, Clustering

Pada *supervsied learning*, kita memprediksi kelas berdasarkan *feature vector* yang merepresentasikan suatu instans (data/observasi). *Feature vector* bisa diibaratkan sebagai sifat-sifat atau keadaan yang diasosiasikan dengan kelas. Pada *supervised learning*, setiap *feature vector* berkorespondisi dengan kelas tertentu. Mencari kelas yang berkorespondensi terhadap suatu *input* disebut **klasifikasi** (***classification***). Contoh klasifikasi adalah mengkategorikan gambar buah (e.g. apel, jeruk, dsb). Sementara itu, apabila kita ingin mencari hubungan antara satu atribut dan atribut lainnya, disebut **association**. Sebagai contoh pada Tabel 3.1, apabila $outlook = sunny$, maka sebagian besar $humidity = high$. Di lain pihak, pada *unsupervised learning* tidak ada kelas yang berkorespondensi; kita mengelompokkan data dengan sifat-sifat yang mirip, disebut ***clustering***. Contoh *clustering* adalah pengelompokkan barang di supermarket. Perlu kamu catat bahwa $unsupervised\ learning \neq clustering$. *Clustering* adalah salah satu *task* pada *unsupervised learning*.

Pada Tabel 3.1, hanya ada dua kelas, klasifikasi data ini disebut ***binary classification***. Apabila kelas klasifikasi lebih dari dua, disebut ***multi-class classification/multi-label classification***. Mohon bedakan antara ***multi-class classification*** dan ***multi-level/hierarchical classification***. Pada *multi-level/hierarchical classification*, pertama-tama kita melakukan klasifikasi untuk suatu kelas generik, lalu dilanjutkan mengklasifikan data ke kelas yang lebih spesifik. Contoh *multi-level classification* adalah *kingdom* (biologi), pertama diklasifikasikan ke *kingdom animalia*, lalu lebih spesifiknya ke *phylum Vertebrata*, dst. *Multi-class/multi-label classification* hanya proses klasifikasi ke dalam banyak "kelas" tanpa tinjauan hirarkis. Klasifikasi yang telah disebutkan sebelumnya disebut juga sebagai *hard classification*, artinya apabila data diklasifikasikan ke kelas tertentu, maka tidak mungkin data berada di kelas lainnya (ya atau tidak). Selain *hard classification*, ada juga yang disebut sebagai *soft classification*, yaitu mengklasifikasikan data ke kelas-kelas

[6] `http://scikit-learn.org/stable/modules/feature_selection.html`

tertentu berdasarkan probabilitas (fuzzy). Misalkan data X memiliki 70% probabilitas sebagai kelas A dan 30% sebagai kelas B.

3.7 Mengukur Kinerja

Pada bab 1, sudah dijelaskan bahwa kita harus mengukur kinerja model dengan cara yang kuantitatif. Salah satu contoh *utility function* yaitu *squared error function* (dijelaskan kemudian pada bab 5). Selain *error function*, kamu juga dapat membandingkan kinerja dengan menggunakan fungsi lainnya seperti akurasi, presisi, *recall*, F1-measure, BLEU [21], ROUGE [22], *intra-cluster similarity*, dsb. Masing-masing *utility function* mengukur hal yang berbeda. Perlu kamu ketahui bahwa memilih ukuran kinerja tergantung pada domain permasalahan. Misalkan pada translasi otomatis, peneliti menggunakan ukuran BLEU; pada peringkasan dokumen, menggunakan ROUGE. Sementara itu, pada *information retrival*/sistem temu balik informasi menggunaan presisi, *recall*, F1-measure, atau *mean average precision* (MAP). Pada domain klasifikasi gambar, menggunakan akurasi. Masing-masing *utility function* dapat memiliki cara mencapai titik optimal yang berbeda. Kamu harus mengerti domain permasalahan untuk mengerti cara mencapai titik optimal. Sebagai pengantar, diktat ini tidak dapat membahas seluruh domain. Dengan demikian, kamu harus membaca lebih lanjut literatur spesifik domain, misal buku pemrosesan bahasa alami atau sistem temu balik informasi, dsb. Sebagai contoh, untuk permasalahan klasifikasi, akurasi sering digunakan. Akurasi didefinisikan pada persamaan 3.1

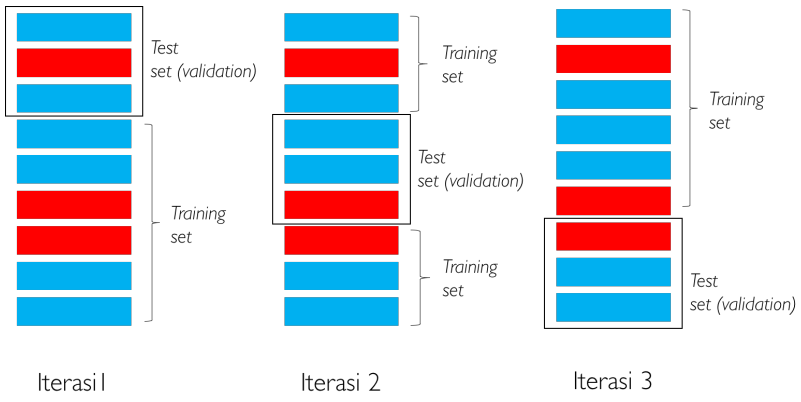
$$\text{akurasi} = \frac{\#\textit{input} \text{ diklasifikasikan dengan benar}}{\text{banyaknya data}} \tag{3.1}$$

3.8 Evaluasi Model

Ada beberapa hal yang perlu kamu catat tentang proses evaluasi suatu model pembelajaran mesin:

1. ***Data splitting***. Seperti yang sudah dijelaskan pada subbab 1.5, pada umumnya kita memiliki *training*, *validation*/*development*, dan *testing data*. Mesin dilatih menggunakan *training data*, saat proses *training*, *performance measure* diukur berdasarkan kemampuan mengenali/ mengeneralisasi *validation data*. Perlu diketahui, *performance measure* diukur menggunakan *validation data* untuk menghindari *overfitting* dan *underfitting*. Setelah selesai dilatih, maka model hasil pembelajaran dievaluasi dengan *testing data*. *Training*, *validation*, dan *testing* data tersusun oleh data yang independen satu sama lain (tidak beririsan) untuk memastikan model yang dihasilkan memiliki generalisasi cukup baik.

2. ***Overfitting dan Underfitting***. *Overfitting* adalah keadaan ketika model memiliki kinerja baik hanya untuk *training data*/*seen examples* tetapi tidak memiliki kinerja baik untuk *unseen examples*. *Underftting* adalah keadaan ketika model memiliki kinerja buruk baik untuk *training data* dan *unseen examples*. Hal ini akan dibahas lebih detil pada subbab 5.7.
3. ***Cross validation***. *Cross validation* adalah teknik untuk menganalisis apakah suatu model memiliki generalisasi yang baik (mampu memiliki kinerja yang baik pada *unseen examples*). Seperti yang kamu ketahui, kita dapat membagi data menjadi *training*, *validation*, dan *testing data*. Saat proses *training*, kita latih model dengan *training data* serta dievaluasi menggunakan *validation data*. Teknik *cross validation* bekerja dengan prinsip yang sama, yaitu membagi sampel asli menjadi beberapa subsampel dengan partisi sebanyak K (K-*fold*). Ilustrasi diberikan oleh Gambar 3.2. Persegi panjang melambangkan suatu instans. Saat proses *training*, kita bagi data menjadi *training data* dan *test data* (i.e., *validation data*). Hal ini diulang sebanyak K kali. Kita evaluasi kemampuan generalisasi model dengan merata-ratakan kinerja pada tiap iterasi. Setelah itu, model dengan kinerja terbaik (pada iterasi teretentu) digunakan lebih lanjut untuk proses *testing* atau dipakai secara praktis. Perlu diperhatikan, setiap subsampel sebaiknya memiliki distribusi yang sama dengan sampel aslinya (keseluruhan sampel); i.e., pada contoh, proporsi warna biru dan merah adalah sama tiap partisi tiap iterasi. Konsep tersebut lebih dikenal dengan ***stratified sampling***[7].



Gambar 3.2. Ilustrasi 3-*fold cross validation*

[7] `https://en.wikipedia.org/wiki/Stratified_sampling`

3.9 Kategori Jenis Algoritma

Algoritma pembelajaran mesin dapat dibagi menjadi beberapa kategori. Dari sudut pandang apakah algoritma memiliki parameter yang harus dioptimasi, dapat dibagi menjadi[8]:

1. Parametrik. Pada kelompok ini, kita mereduksi permasalahan sebagai optimisasi parameter. Kita mengasumsikan permasalahan dapat dilambangkan oleh fungsi dengan bentuk tertentu (e.g., linear, polinomial, dsb). Contoh kelompok ini adalah model linear.
2. Non parametrik. Pada kelompok ini, kita tidak mengasumsikan permasalahan dapat dilambangkan oleh fungsi dengan bentuk tertentu. Contoh kelompok ini adalah *Naive Bayes*, *decision tree* (ID3) dan *K-Nearest Neighbors*.

Dari sudut pandang lainnya, jenis algoritma dapat dibagi menjadi:

1. Model linear, contoh regresi linear, regresi logistik, *support vector machine*.
2. Model probabilistik, contoh *Naive Bayes*, *hidden markov model*.
3. Model non-linear, yaitu (*typically*) *artificial neural network*.

Selain kedua skema pengkategorian ini, terdapat skema pengkategorian lain (silahkan eksplorasi sendiri).

3.10 Tahapan Analisis

Bagian ini adalah ringkasan bab ini. Untuk menganalisis data, terdapat langkah yang perlu kamu perhatikan

1. Memutuskan tujuan analisis data (*defining goal*)
2. Mendapatkan data
3. Merapihkan data
4. Merepresentasikan data sebagai *feature vector*
5. Melakukan transformasi dan/atau *feature selection* (mengurasi dimensi *feature vector*)
6. Melatih model (*training*)
7. Melakukan *testing* dan analisis model baik secara kuantitatif dan kualitatif
8. Menyajikan data (presentasi)

[8] *Artificial Neural Network* dapat dikategorikan ke dalam keduanya.

Soal Latihan

3.1. Konversi atribut
Sebutkan dan jelaskan macam-macam cara untuk mengkonversi atribut! Sebagai contoh, numerik-nominal dan nominal-numerik.

3.2. Transformasi data
Sebutkan dan jelaskan macam-macam cara transformasi data (e.g. merubah *non-linearly separable* menjadi *linearly separable*)

3.3. Seleksi fitur
Bacalah algoritma seleksi fitur pada *library* sklearn. Jelaskan alasan (*rationale*) dibalik penggunaan tiap algoritma yang ada!

3.4. Inductive Learning
Jelaskanlah algoritma *list-then-eliminate* dan *candidate-elimination*!