# Part 8 - When Model Meets Data

Bab ini menjembatani konsep-konsep matematika dasar yang telah diperkenalkan di bagian pertama buku dengan empat pilar utama dalam *machine learning*: Regresi (Bab 9), *Dimentionality Reduction* (Bab 10), Estimasi Kepadatan (Bab 11), dan Klasifikasi (Bab 12). Tujuannya adalah untuk mendemonstrasikan bagaimana konsep-konsep matematika tersebut digunakan untuk merancang algoritma *machine learning*. Bab ini akan menguraikan tiga komponen utama dari sistem *machine learning*: data, model, dan pembelajaran, serta memperkenalkan kerangka kerja untuk melatih model agar dapat berkinerja baik pada data yang belum pernah dilihat sebelumnya.

# 8.1 Data, Model, dan Pembelajaran

Prinsip utama dalam *machine learning* adalah bahwa model yang baik harus berkinerja baik pada data yang belum pernah dilihat (*unseen data*). Ini memerlukan metrik kinerja dan cara untuk mencapai kinerja yang baik di bawah metrik tersebut.

## 8.1.1 Data sebagai Vektor

| Name | Gender | Degree | Postcode | Age | Annual salary |
|---|---|---|---|---|---|
| Aditya | M | MSc | W21BG | 36 | 89563 |
| Bob | M | PhD | EC1A1BA | 47 | 123543 |
| Chloé | F | BEcon | SW1A1BH | 26 | 23989 |
| Daisuke | M | BSc | SE207AT | 68 | 138769 |
| Elisabeth | F | MBA | SE10AA | 33 | 113888 |

Data diasumsikan berbentuk tabel, di mana setiap baris adalah sebuah *contoh* atau *titik data* (*data point*) dan setiap kolom adalah sebuah *fitur* (*feature*). Data non-numerik, seperti jenis kelamin atau kode pos, harus diubah menjadi representasi numerik yang sesuai.

| Gender ID | Degree | Latitude (in degrees) | Longitude (in degrees) | Age | Annual Salary (in thousands) |
|---|---|---|---|---|---|
| -1 | 2 | 51.5073 | 0.1290 | 36 | 89.563 |
| -1 | 3 | 51.5074 | 0.1275 | 47 | 123.543 |
| +1 | 1 | 51.5071 | 0.1278 | 26 | 23.989 |
| -1 | 1 | 51.5075 | 0.1281 | 68 | 138.769 |
| +1 | 2 | 51.5074 | 0.1278 | 33 | 113.888 |

*Note:* Kolom nama dihapus karena tidak penting bagi proses pembelajaran dan juga untuk *privacy concern.*
Selanjutnya, kita asumsikan setiap masukan $x_n$ adalah vektor berdimensi $D$ yang berisi bilangan riil. Sebuah dataset sering kali direpresentasikan sebagai sekumpulan pasangan contoh-label $(x_1, y_1), \ldots, (x_N, y_N)$, di mana matriks contoh ditulis sebagai $x \in \mathbb{R}^{N \times D}$ dan $y_n$ adalah label atau target. Representasi vektor ini memungkinkan kita menggunakan konsep dari aljabar linear, geometri, dan optimisasi.

## 8.1.2 Model sebagai Fungsi

Setelah data direpresentasikan dalam bentuk vektor, kita dapat membangun fungsi prediktif (dikenal sebagai *prediktor*). Sebuah prediktor adalah fungsi yang memetakan contoh masukan (vektor fitur) ke sebuah keluaran, umumnya skalar bernilai real. Ini dapat ditulis sebagai: $f : \mathbb{R}^D \to \mathbb{R}$. Di sini bahasan berfokus pada fungsi linear (afin) yang dapat dinyatakan sebagai: $f(x) = \boldsymbol{\theta}^\top x + \theta_0$ di mana $\boldsymbol{\theta}$ dan $\theta_0$ adalah parameter yang tidak diketahui.

## 8.1.3 Model sebagai Distribusi Probabilitas

Data sering dianggap sebagai observasi yang mengandung *noise* dari suatu efek yang mendasarinya. Teori probabilitas menyediakan bahasa untuk mengukur ketidakpastian ini. Alih-alih satu fungsi tunggal, prediktor dapat dianggap sebagai model probabilistik yang menggambarkan distribusi dari fungsi-fungsi yang mungkin. Hal ini memungkinkan kita untuk mengukur kepercayaan terhadap prediksi yang dihasilkan, sering kali digambarkan sebagai distribusi Gaussian di sekitar nilai prediksi.

## 8.1.4 Pembelajaran adalah Tentang Menemukan Parameter

Tujuan dari pembelajaran adalah menemukan parameter model agar prediktor yang dihasilkan berkinerja baik pada data yang belum pernah dilihat. Ada tiga fase algoritma yang berbeda:

1. Prediksi atau inferensi: Menggunakan prediktor yang telah dilatih pada data uji baru.
2. Pelatihan atau estimasi parameter: Menyesuaikan model prediktif berdasarkan data pelatihan.
3. Penalaan hiperparameter atau seleksi model: Membuat keputusan pemodelan tingkat tinggi, seperti memilih struktur prediktor. Untuk menghindari *overfitting* (model terlalu

cocok dengan data pelatihan tetapi tidak dapat menggeneralisasi), kita menggunakan teknik seperti *regularisasi* atau menambahkan *prior*.

# 8.2 Minimisasi Risiko Empiris (Empirical Risk Minimization - ERM)

ERM adalah strategi umum untuk pembelajaran yang tidak secara eksplisit membangun model probabilistik. Proses ini melibatkan empat pilihan desain utama.

## 8.2.1 Kelas Hipotesis Fungsi

Langkah pertama adalah menentukan himpunan fungsi yang diizinkan untuk menjadi prediktor, yang disebut kelas hipotesis. Sebagai contoh, kita dapat memilih kelas fungsi afin, yang dapat ditulis secara ringkas sebagai $f(x_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top x_n$ dengan menambahkan fitur konstan ke $x_n$.

## 8.2.2 Fungsi Kerugian untuk Pelatihan

Untuk mengukur seberapa baik model cocok dengan data, kita mendefinisikan *fungsi kerugian* (loss function) $\ell(y_n, \hat{y}_n)$ yang menghasilkan nilai non-negatif yang merepresentasikan kesalahan prediksi. Tujuan kita adalah meminimalkan *risiko empiris*, yaitu rata-rata kerugian pada seluruh data pelatihan:

$$R_{\text{emp}}(f, x, \mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n)$$

di mana $\hat{y}_n = f(x_n, \boldsymbol{\theta})$. Sebagai contoh, dengan *squared loss*

$$\ell(y_n, \hat{y}_n) = (y_n - \hat{y}_n)^2$$

dan prediktor linear, kita mendapatkan masalah *least-squares*:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} (y_n - \boldsymbol{\theta}^\top x_n)^2$$

atau dalam bentuk matriks $\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} |\mathbf{y} - x\boldsymbol{\theta}|^2$.

## 8.2.3 Regularisasi untuk Mengurangi Overfitting

Minimisasi risiko empiris dapat menyebabkan *overfitting*, di mana prediktor terlalu cocok dengan data pelatihan dan gagal menggeneralisasi ke data baru. Untuk mengatasinya,

ditambahkan sebuah istilah penalti yang disebut *regularisasi*. Ini menyeimbangkan antara meminimalkan kerugian pada data pelatihan dan kompleksitas solusi. Contohnya adalah *regularized least squares*:

$$\min_{\boldsymbol{\theta}} \frac{1}{N}|\mathbf{y} - x\boldsymbol{\theta}|^2 + \boxed{\lambda|\boldsymbol{\theta}|^2}$$

di mana $\lambda$ adalah parameter regularisasi.

### 8.2.4 Validasi Silang (Cross Validation) untuk Menilai Kinerja Generalisasi

Untuk mengestimasi kinerja generalisasi model pada data yang terbatas, digunakan *validasi silang K-fold cross-validation*. Data dibagi menjadi $K$ bagian; $K-1$ bagian digunakan untuk pelatihan, dan satu bagian digunakan sebagai set validasi. Proses ini diulang $K$ kali, dan rata-rata kinerja dari $K$ putaran tersebut dihitung untuk mendapatkan estimasi yang lebih stabil dari kesalahan generalisasi.

## 8.3 Estimasi Parameter

Bagian ini membahas pendekatan probabilistik untuk pembelajaran, di mana kita menggunakan distribusi probabilitas untuk memodelkan ketidakpastian.

### 8.3.1 Estimasi Kemungkinan Maksimum (Maximum Likelihood Estimation - MLE)

Ide di balik MLE adalah menemukan parameter $\boldsymbol{\theta}$ yang membuat data yang diamati menjadi paling mungkin. Ini dilakukan dengan memaksimalkan fungsi *likelihood* $p(\mathbf{Y}|x, \boldsymbol{\theta})$, yang setara dengan meminimalkan *negatif log-likelihood*:

$$L(\boldsymbol{\theta}) = -\log p(\mathbf{Y}|x, \boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n|x_n, \boldsymbol{\theta})$$

Jika kita mengasumsikan *likelihood* Gaussian $p(y_n|x_n, \boldsymbol{\theta}) = \mathcal{N}(y_n|x_n^\top \boldsymbol{\theta}, \sigma^2)$, maka minimisasi negatif log-likelihood akan setara dengan masalah *least-squares*.

### 8.3.2 Estimasi Maximum A Posteriori (MAP)

Jika kita memiliki pengetahuan sebelumnya tentang parameter, kita dapat memasukkannya sebagai distribusi *prior* $p(\boldsymbol{\theta})$. Dengan menggunakan Teorema Bayes, kita dapat menghitung

distribusi *posterior*:

$$p(\boldsymbol{\theta}|x) \propto p(x|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

Estimasi MAP mencari parameter $\boldsymbol{\theta}$ yang memaksimalkan probabilitas posterior ini. Pendekatan ini analog dengan regularisasi dalam ERM, di mana prior berfungsi sebagai penalti terhadap parameter yang kompleks.

### 8.3.3 Pencocokan Model (Model Fitting)

Saat mencocokkan model dengan data, ada tiga kemungkinan hasil:

1. *Overfitting*: Kelas model terlalu kaya atau kompleks, sehingga model menangkap *noise* dalam data pelatihan dan tidak dapat menggeneralisasi dengan baik.
2. *Underfitting*: Kelas model tidak cukup kaya untuk menangkap struktur yang mendasari dalam data.
3. Cocok dengan baik: Kelas model memiliki kompleksitas yang tepat untuk data yang diberikan.

## 8.4 Pemodelan Probabilistik dan Inferensi

Pendekatan ini menggunakan probabilitas untuk memodelkan ketidakpastian dalam data dan parameter model secara konsisten.

### 8.4.1 Model Probabilistik

Sebuah model probabilistik ditentukan oleh distribusi gabungan (*joint distribution*) dari semua variabel acak yang terlibat, baik yang diamati maupun yang tersembunyi.

### 8.4.2 Inferensi Bayesian

Berbeda dari MLE atau MAP yang menghasilkan satu estimasi titik terbaik untuk parameter, inferensi Bayesian bertujuan untuk menemukan seluruh distribusi posterior parameter, $p(\boldsymbol{\theta}|x)$. Prediksi dibuat dengan mengintegrasikan (memarginalkan) semua nilai parameter yang mungkin:

$$p(x) = \int p(x|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} = \mathbb{E}_{\boldsymbol{\theta}}[p(x|\boldsymbol{\theta})]$$

Tantangan komputasi utama dalam inferensi Bayesian adalah integrasi, bukan optimisasi seperti pada MLE/MAP.

### 8.4.3 Model Variabel Laten

Terkadang, model menyertakan *variabel laten* $\mathbf{z}$ tambahan yang tidak diamati secara langsung. Variabel ini dapat menyederhanakan struktur model atau merepresentasikan proses tersembunyi yang menghasilkan data. Untuk melakukan inferensi, variabel laten ini harus dimarginalkan untuk mendapatkan *likelihood*:

$$p(x|\boldsymbol{\theta}) = \int p(x|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z}) d\mathbf{z}$$

## 8.5 Model Grafis Terarah (Directed Graphical Models)

Model grafis terarah, juga dikenal sebagai *Jaringan Bayesian* (*Bayesian Networks*), menyediakan bahasa visual yang ringkas untuk menentukan model probabilistik dan dependensi antar variabel acak.

### 8.5.1 Semantik Graf

Dalam model grafis, simpul (nodes) merepresentasikan variabel acak, dan panah (edges) merepresentasikan hubungan probabilitas bersyarat. Struktur graf menentukan faktorisasi dari distribusi gabungan. Untuk satu set variabel $x = x_1, \ldots, x_K$, distribusi gabungannya adalah:

$$p(x) = \prod_{k=1}^{K} p(x_k|\mathrm{Pa}_k)$$

di mana $\mathrm{Pa}_k$ adalah himpunan simpul induk (parent) dari $x_k$. *Notasi lempeng* (*plate notation*) digunakan untuk merepresentasikan variabel yang diulang secara ringkas.

### 8.5.2 Independensi Bersyarat dan d-Separation

Model grafis memungkinkan kita untuk menyimpulkan properti independensi bersyarat dari distribusi gabungan hanya dengan memeriksa struktur graf. Konsep yang disebut *d-separation* digunakan untuk menentukan apakah himpunan simpul $A$ independen secara bersyarat dari himpunan $B$ diberikan himpunan $C$, yang dinotasikan sebagai $A \perp B | C$.

## 8.6 Seleksi Model

Seleksi model adalah proses memilih keputusan pemodelan tingkat tinggi (misalnya, jenis model atau nilai hiperparameter) yang secara kritis memengaruhi kinerja.

## 8.6.1 Validasi Silang Bersarang (Nested Cross-Validation)

Ini adalah pendekatan dua tingkat untuk seleksi model. Tingkat dalam (*inner loop*) dari validasi silang digunakan untuk memilih model atau hiperparameter terbaik dengan mengevaluasi kinerja pada set validasi. Tingkat luar (*outer loop*) kemudian memberikan estimasi kinerja generalisasi yang tidak bias dari model yang telah dipilih pada set uji.

## 8.6.2 Seleksi Model Bayesian

Pendekatan Bayesian untuk seleksi model secara otomatis menyeimbangkan antara kecocokan data dan kompleksitas model, sebuah konsep yang dikenal sebagai *Pisau Cukur Occam* (*Occam's razor*). Model dibandingkan berdasarkan *bukti model* (*model evidence*) atau *marginal likelihood*, $p(\mathbf{D}|M_k)$, yang dihitung dengan mengintegrasikan semua parameter model:

$$p(\mathbf{D}|M_k) = \int p(\mathbf{D}|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k|M_k)d\boldsymbol{\theta}_k$$

Model dengan bukti tertinggi yang dipilih.

## 8.6.3 Faktor Bayes untuk Perbandingan Model

Untuk membandingkan dua model, $M_1$ dan $M_2$, kita dapat menghitung *Faktor Bayes* (*Bayes factor*), yang merupakan rasio dari bukti model mereka:

$$\frac{p(\mathbf{D}|M_1)}{p(\mathbf{D}|M_2)}$$

Rasio ini mengukur seberapa baik data $\mathbf{D}$ diprediksi oleh $M_1$ dibandingkan dengan $M_2$. Jika rasio lebih besar dari 1, model $M_1$ lebih disukai.