BAB 2

LANDASAN TEORI

2.1. Metagenomika

Metagenomika merupakan sebuah cabang ilmu yang mempelajari materi genetik pada sampel yang diambil secara langsung dari lingkungan (Simangunsong, 2015). Tidak seperti ilmu genomika yang di mana analisisnya dilakukan hanya terhadap beberapa organisme tertentu yang terisolasi (atau dikultur sebelumnya), analisis metagenomik dilakukan terhadap sekumpulan komunitas mikroorganisme secara langsung tanpa memerlukan upaya pengkulturan dahulu. Hal ini telah membawa wawasan menarik terkait sistem ekologi dari berbagai habitat (Richter dkk., 2008). Kemunculan bidang ilmu penelitian ini dipicu oleh pengembangan dan perkembangan terhadap teknologi *sequencing*, dengan teknik *454 pyrosequencing* oleh Roche sebagai salah satu contohnya (Margulies dkk., 2005).

2.2. Machine Learning

Machine learning, atau disebut juga pembelajaran mesin, adalah sebuah bidang ilmu yang mempelajari tentang algoritma komputer yang berkembang secara mandiri melalui pengalaman (Mitchell, 1997). Model yang dibangun untuk pembelajaran mesin bekerja dengan menggunakan masukan berupa data latih (*training set*) sebagai bahan pembelajaran untuk melakukan analisis terhadap data uji (*test set*). Bentuk pembelajaran yang umum diterapkan dalam merancang

model pembelajaran mesin terdiri dari *supervised learning* dan *unsupervised learning*. *Supervised learning* mengacu pada pembelajaran yang dalam data latihnya terdapat informasi *input* beserta *output* yang bersangkutan, sedangkan dalam *unsupervised learning*, data latih yang digunakan hanya mengandung informasi *input*, sehingga model harus menarik kesimpulan (*output*) sendiri berdasarkan *input* yang dipelajari (Bishop, 2006).

2.3. Linear Discriminant Analysis

Linear discriminant analysis (LDA) adalah salah satu metode reduksi dimensi data yang memilih dan mengekstraksi kumpulan ciri yang paling menonjol (diskriminatif) dari data untuk klasifikasi lebih dari satu kelas (Yan dkk., 2020). Metode ini termasuk sebagai salah satu yang paling berguna dan populer dengan berbagai penerapannya, termasuk dalam ranah *clustering* (Liu dkk., 2020). Perbedaan utama LDA dengan metode reduksi dimensi lain yang juga kerap digunakan, *principal component analysis* (PCA), terletak pada fokus utama masing-masing metode. LDA dan PCA bertujuan mencari komponen dengan variansi tertinggi dalam data, namun LDA juga mengutamakan tingkat keterpisahan antar kelas data (Raschka and Mirjalili, 2019).

Menurut Raschka dan Mirjalili (2019), metode *linear discriminant analysis* ini dapat diringkas ke dalam tujuh langkah berikut:

1. Standarisasikan himpunan data awal berdimensi *d*, di mana *d* menyatakan banyaknya ciri/fitur dalam data.
2. Hitung vektor rata-rata (*mean*) dimensi *d* untuk setiap kelas pada data.

3. Buat matriks persebaran antar-kelas (*between-class scatter matrix*; S_B) dan matriks persebaran dalam-kelas (*within-class scatter matrix*; S_W).
4. Hitung kumpulan *eigenvector* beserta *eigenvalue* yang berkaitan dari matriks $S_W^{-1}S_B$.
5. Urutkan kumpulan *eigenvalue* yang ada dari nilai terbesar ke terkecil (*descending*).
6. Ambil sebanyak *k eigenvector* dengan *eigenvalue* terbesar untuk membentuk matriks transformasi *W* yang berdimensi $d \times k$, di mana setiap *eigenvector* mewakili satu kolom.
7. Gunakan matriks transformasi *W* tersebut untuk mentransformasikan matriks data awal *X* berdimensi *d* yang diteliti ke dalam matriks fitur baru berdimensi *k*.

2.3.1. Normalisasi Data

Data fitur yang sudah diekstraksi terlebih dahulu dinormalisir dengan metode *min-max scaling* ke dalam rentang nilai [0, 1]. Normalisasi ini diterapkan terhadap setiap fitur dengan rumus seperti Rumus 2.1 di bawah ini (Raschka and Mirjalili, 2019):

$$x_{normal} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.1}$$

Dalam setiap fitur, nilai hasil normalisasi x_{normal} diperoleh dengan menghitung bobot nilai individu *x* terhadap rentang nilai dalam fitur tersebut, yakni selisih antara nilai terbesar dan terkecil x_{max} dan x_{min}.

2.3.2. Matriks Persebaran

Untuk memperoleh kedua matriks persebaran *within-class* dan *between-class*, perlu dilakukan terlebih dahulu perhitungan untuk vektor rata-rata (*mean vector*) setiap kelas. Tiap *mean vector* m_i memuat nilai rata-rata ciri/fitur (*mean feature value*) μ_m untuk setiap kelas *i*:

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x_m \tag{2.2}$$

Rumus 2.2 di atas akan menghasilkan sebanyak *c* vektor rata-rata, di mana *c* menyatakan banyaknya kelas dalam himpunan data:

$$m_i = \begin{bmatrix} \mu_{i(\text{feature } 1)} \\ \mu_{i(\text{feature } 2)} \\ \vdots \\ \mu_{i(\text{feature } n)} \end{bmatrix}^T, i \in \{1, 2, \dots, c\} \tag{2.3}$$

Dengan kumpulan *mean vector* tersebut, *within-class scatter matrix* kemudian dapat dihitung dengan menjumlahkan *scatter matrix* dari setiap kelas *i*:

$$S_W = \sum_{i=1}^{c} S_i \tag{2.4}$$

$$S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^T \tag{2.5}$$

Namun demikian, dalam komputasi matriks persebaran, LDA tunduk terhadap asumsi di mana data dalam himpunan latih (*training set*) terdistribusi secara normal sebelum pengerjaan. Agar asumsi ini tetap terpenuhi semaksimal mungkin, *scatter matrix* dari setiap kelas *i* (S_i) dapat diskala terlebih dahulu sebelum dijumlahkan menjadi *within-class scatter matrix* (S_W). Ketika setiap *scatter matrix* dibagi dengan banyaknya sampel setiap kelas (n_i), dapat dilihat

bahwa perhitungan *within-class scatter matrix* ini sama dengan perhitungan *covariance matrix* (Σ_i), versi normalisasi dari matriks persebaran tersebut:

$$\Sigma_i = \frac{1}{n_i} S_i = \frac{1}{n_i} \sum_{x \in D_i} (x - m_i)(x - m_i)^T \quad (2.6)$$

Setelah matriks persebaran *within-class* (atau matriks kovariansi) selesai dihitung, matriks persebaran *between-class* (S_B) dapat dihitung dengan Rumus 2.7 di bawah:

$$S_B = \sum_{i=1}^{c} n_i \, (m_i - m)(m_i - m)^T \quad (2.7)$$

Variabel *m* merupakan rata-rata seluruh kelas (*total mean*/*overall mean*) dari sebanyak *c* kelas yang ada dalam data.

2.3.3. Eigenvector dan Eigenvalue

Setelah kedua matriks persebaran *within-class* dan *between-class* selesai dikomputasi, *eigenvector* dan *eigenvalue* dari masing-masing matriks persebaran tersebut dapat diperoleh dengan penyelesaian persamaan matriks $S_W^{-1} S_B$.

2.3.4. Transformasi

Dengan nilai-nilai *eigenvalue* terbesar yang sudah diperoleh, kumpulan *eigenvector* yang berkaitan kemudian digabungkan menjadi kolom-kolom untuk matriks transformasi *W*. Proses transformasi kemudian dilakukan dengan perkalian matriks seperti pada Rumus 2.8 di bawah ini, di mana *X* merupakan matriks himpunan latih dan *X'* matriks baru:

$$X' = XW \quad (2.8)$$

2.4. Hierarchical Clustering

Hierarchical clustering merupakan salah satu metode statistika untuk melakukan *clustering*. Metode ini bekerja dengan terlebih dahulu mengukur tingkat ketidaksamaan antar dua gugusan data sebelum menggabungkan kedua gugusan tersebut menjadi satu gugusan baru (*agglomerative*) atau memecah masing-masing gugusan menjadi dua gugusan baru (*divisive*). Keunggulan dari metode ini termasuk di antaranya kemudahan dalam pemahaman maupun penerapannya serta tidak adanya kewajiban untuk terlebih dahulu mengetahui jumlah *cluster* yang diinginkan (Sultana, 2020). Perbedaan antara metode *agglomerative* dan *divisive* terletak pada alur pengerjaannya, di mana *agglomerative hierarchical clustering* membentuk gugusan besar data dengan menggabungkan sampel-sampel individual atau kumpulan gugusan kecil (*bottom-up*), sedangkan *divisive hierarchical clustering* membentuk kumpulan gugusan kecil data dengan memecah gugusan data yang lebih besar (Kaufman and Rousseeuw, 1990). Metode *hierarchical clustering* yang akan digunakan dalam penelitian ini adalah metode *agglomerative*.

Tingkat ketidaksamaan ini diukur berdasarkan metrik jarak antara dua gugus data, dan penggabungan atau pemecahan kedua gugus tersebut dilakukan berdasarkan kriteria penghubungan tertentu (*linkage criteria*). Ada beberapa macam metrik dan kriteria yang umum digunakan dalam penelitian *hierarchical clustering*, termasuk metrik jarak Euclidean (Rumus 2.9) dan kriteria

penghubungan *single-linkage* (disingkat SLINK; Rumus 2.10), yang diperoleh dari sumber panduan pengguna milik SAS Institute (SAS Institute Inc., 2009):

$$\|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2} \tag{2.9}$$

$$\min\{d(a,b) \mid a \in A, b \in B\} \tag{2.10}$$

2.5. Silhouette Index

Proses peninjauan kinerja model yang menggunakan algoritma *clustering* berbeda dengan peninjauan kinerja algoritma klasifikasi pada umumnya. Dalam pengerjaan *clustering*, kualitas yang diukur adalah kedekatan antar sampel dalam satu gugusan yang sama, keterpisahan satu sampel dari sampel-sampel serupa yang merupakan bagian dari gugusan yang berbeda, serta keterpisahan antar gugusan (Pedregosa *et al.*, 2011). Untuk meninjau keabsahan hasil dari algoritma yang demikian, terdapat beberapa sistem tolak ukur berbeda yang dapat digunakan, dan sistem yang digunakan dalam penelitian ini adalah *silhouette index/coefficient*.

Silhouette index bekerja dengan membandingkan rata-rata jarak suatu sampel *i* terhadap semua sampel lainnya dalam satu gugusan yang sama, C_i, dengan jarak terdekat sampel tersebut terhadap semua *gugusan* lainnya, C_k. Rumus dari *silhouette index* dinyatakan sebagai berikut.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, if\ |C_i| > 1 \tag{2.11}$$

Dari Rumus 2.11 di atas, $a(i)$ menyatakan rata-rata jarak sampel individu *i* terhadap setiap sampel lain *j* dalam gugusan C_i, $b(i)$ menyatakan jarak sampel *i*

terhadap gugusan lain C_k yang terdekat (tampak dari operator *min* dalam rumus), dan $s(i)$ menyatakan nilai *silhouette index* untuk sampel i. Dalam kasus di mana sebuah gugusan C_i hanya memiliki i sebagai sampel tunggal di dalamnya, maka $s(i)$, *silhouette index* untuk sampel i tersebut, akan bernilai nol. Adapun kedua jarak sampel $a(i)$ dan $b(i)$ dinyatakan dalam Rumus 2.12 dan 2.13 sebagai berikut.

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \tag{2.12}$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j) \tag{2.13}$$