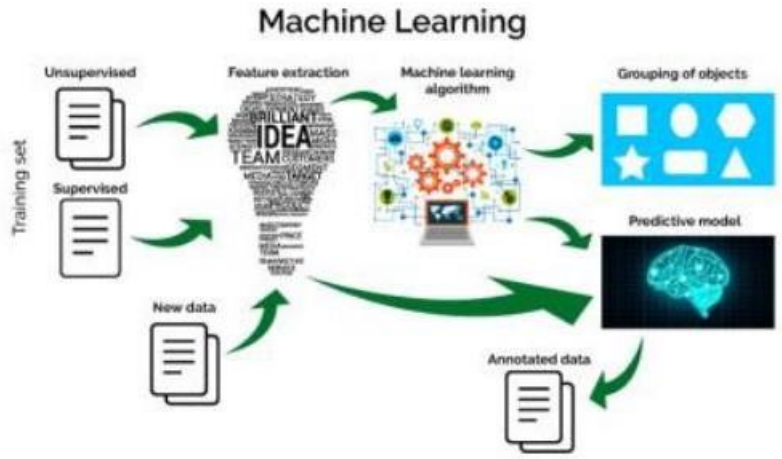
BAB II
TINJAUAN PUSTAKA

2.1 Machine Learning

Machine learning merupakan serangkaian teknik yang dapat membantu dalam menangani dan memprediksi data yang sangat besar dengan cara mempresentasikan data-data tersebut dengan algoritma pembelajaran (Danukusumo, 2017).



Gambar 2.1 Gambar Machine Learning (Pantech, 2018)

Istilah machine learning sendiri pertama kali didefinisikan oleh Arthur Samuel pada tahun 1959. Menurut Arthur Samuel, machine learning adalah suatu bidang ilmu komputer yang memberikan kemampuan pembelajaran kepada komputer untuk mengetahui sesuatu tanpa pemrograman yang jelas.

Menurut (Mohri et.al, 2012) machine learning dapat didefinisikan sebagai metode komputasi berdasarkan pengalaman untuk meningkatkan performa atau membuat prediksi yang akurat. Definisi pengalaman disini ialah informasi sebelumnya yang telah tersedia dan bisa dijadikan data pembelajar.

2.1.1 Cara Kerja Machine Learning

Secara sederhana, proses dari *machine learning* dibagi menjadi 3 bagian, yaitu: *input* data, abstraksi data, dan generalisasi. Untuk tahapan-tahapan atau alur kerja dari *machine learning* adalah sebagai berikut :

a. Mengumpulkan Data

   Data yang digunakan beragam, mulai dari text dengan formal *file* seperti .csv hingga berbentuk gambar seperti .jpg. Langkah ini merupakan dasar dari pembelajaran. Semakin banyak variasi, kepadatan dan volume data yang relevan, semakin baik prospek pembelajaran untuk mesin.

b. Mempersiapkan Data

   Setiap proses analitis berkembang dengan kualitas data yang digunakan. Pada tahapan ini, data dipersiapkan sedemikian rupa hingga memenuhi apa saja yang dibutuhkan. Tahapan ini diawali dengan analisis data untuk dapat menentukan kualitas data, kemudian melakukan perbaikan-perbaikan pada masalah yang ditemukan seperti hilangnya data dan lain-lain. Tahapan ini biasa disebut juga dengan *pre-processing*.

c. Melatih Sebuah Model

   Pada tahapan ini melibatkan pemilihan algoritma dan data yang tepat dalam bentuk model. Data yang sudah disiapkan sebelumnya kemudian akan dibagi (split) menjadi dua bagian, yaitu: data latih (train) dan data uji (test).Bagian data latih (*training* data) akan digunakan untuk pengembangan model. Sedangkan bagian data uji (*testing* data) akan digunakan sebagai referensi.

d. Mengevaluasi Model

   Untuk menguji seberapa akurat model yang dibuat, bagian data uji (*testing* data) digunakan. Pada tahap ini akan ditentukan ketepatan dalam pemilihan algoritma berdasarkan dari hasil pengujian. Pengujian untuk ketepatan

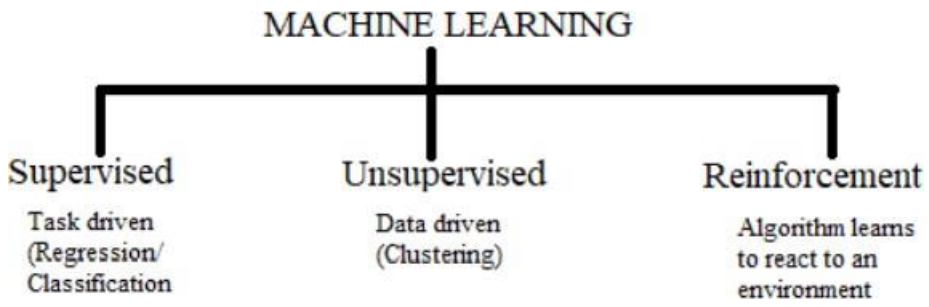
model ini dilakukan dengan cara melihat kinerja pada data yang tidak digunakan sama sekali selama pembuatan model.

e. Meningkatkan Kinerja

Tahap ini bisa jadi akan melibatkan pemilihan model yang lain, atau bisa juga memperkenalkan lebih banyak variabel untuk meningkatkan efisiensi.

2.1.2 Macam-Macam Algoritma Machine Learning

Secara umum algoritma *machine learning* dibagi menjadi 3 jenis, antara lain *Supervised* atau terarah, *Unsupervised* atau tak terarah, dan *Reinforcement* atau bala bantuan. Ada juga yang disebut dengan *Semi-supervised* yang mana merupakan gabungan dari algoritma *supervised* dan *unsupervised.* Untuk lebih jelasnya, dapat dilihat pada gambar 2.2 dibawah ini.



Gambar 2.2 Gambar Jenis-jenis Algoritma *Machine Learning*

1. Supervised Learning

Penggunaan skenario *supervised learning*, pembelajaran menggunakan masukan data pembelajaran yang telah diberi label. Setelah itu membuat prediksi dari data yang telah diberi label.

2. Unsupervised Learning

Penggunaan skenario *unsupervised learning*, pembelajaran menggunakan masukan data pembelajaran yang tidak diberi label. Setelah itu mencoba untuk mengelompokkan data berdasarkan karakteristik-karakteristik yang ditemui.

3. Reinforcement Learning

Pada skenario *reinforcement learning* fase pembelajaran dan tes saling dicampur. Untuk mengumpulkan informasi pembelajar secara aktif dengan berinteraksi ke lingkungan sehingga untuk mendapatkan balasan untuk setiap aksi dari belajar.

2.2 Exploratory Data Analysis

Exploratory Data Analysis merupakan pendekatan yang digunakan untuk menganalisis kumpulan data agar mendapatkan pengetahuan dari dataset itu sendiri sehingga dapat memperkuat hipotesis awal yang mendukung penelitian. Analisis data yang dilakukan adalah seperti melihat sebaran titik data, melihat distribusi data, analisis *univariate*, melihat *skewness* dan *kurtosis* masing-masing atribut, dan melihat korelasi antar atribut. Analisis tersebut dilakukan dengan menggunakan metode visual. Analisis ini juga berguna sebagai gambaran mengenai hal apa yang dapat dilakukan pada saat data *preprocessing*.

Proses peninjauan penyebaran titik data merupakan salah satu proses *exploratory data analysis* yang dilakukan dalam penelitian ini. Proses tersebut melihat penyebaran titik data jika ditinjau menggunakan 2 variabel, contohnya variabel *glucose* dan *blood_pressure*. Ditinjau dari dua variabel tersebut, kecendrungan seseorang terkena penyakit diabetes semakin besar ketika keduanilai variabel tersebut juga besar.

2.3 Data Pre-Processing

Pre-processing data atau pembersihan data merupakan tahapan awal yang dilakukan sebelum masuk ke tahap pembentukan model *machine learning*. Pembersihan data berisi berbagai proses yang bertujuan untuk melakukan perbaikan pada data yang akan diteliti. Melakukan pembersihan data adalah salah satu hal yang penting, karena data yang masih mentah cenderung tidak siap dan perlu dikaji sedemikian rupa terlebih dahulu agar sesuai dengan kebutuhan sistem. Pada pembersihan data ini, kasus yang sering dijumpai adalah hilangnya sebagian

data atau *missing values*. Maksud dari *missing values* adalah adanya beberapa fitur yang tidak memiliki nilai informasi atau kosong yang dimana seharusnya memiliki nilai.

2.4 Data Training dan Data Testing

Menurut Prasetyo (2014), data untuk pengujian klasifikasi dibagi menjadi data training dan data testing. Data atau vektor yang sudah diketahui sebelumnya untuk label kelompok dan digunakan untuk membangun model *classifier* disebut data training. Data atau vektor yang belum diketahui (dianggap belum diketahui) label kelompoknya sehingga menggunakan model *classifier* yang sudah pernah dibangun untuk diketahui label kelompoknya disebut data testing.

Data training digunakan oleh algoritma klasifikasi untuk membentuk sebuah model *classifier*. Model ini merupakan representasi pengetahuan yang akan digunakan untuk memprediksi kelompok dari data baru yang belum pernah ada. Data testing digunakan untuk mengukur sejauh mana *classifier* berhasil melakukan klasifikasi dengan benar. Data yang ada pada data *testing* seharusnya tidak boleh ada pada data *training* sehingga dapat diketahui apakah model *classifier* sudah tepat atau belum dalam melakukan klasifikasi (Witten dan Eibe, 2011).

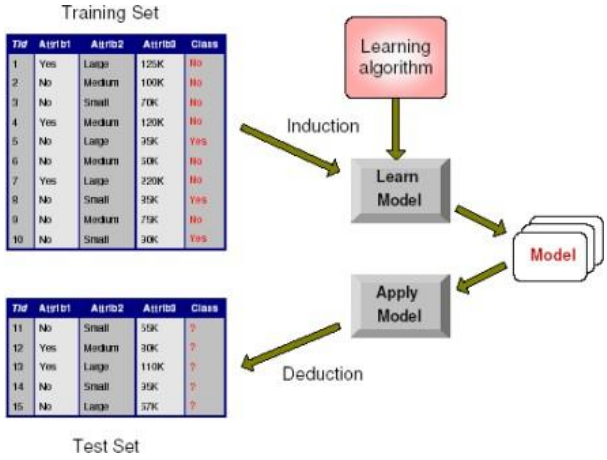
Pembagian data *training* dan data *testing* dengan metode proporsi adalah metode yang paling sederhana namun memiliki beberapa keterbatasan. Jumlah data *training* lebih sedikit yang tersedia untuk pelatihan karena sebagian harus digunakan untuk data *testing*. Akibatnya, model yang dibangun kemungkinan tidak sebaik ketika semua data digunakan sebagai data *training*. Model yang dibangun juga tergantung pada komposisi pemecahan *set* dan *training* dan data *testing*. Semakin sedikit data *training*, maka varian dalam model semakin besar (Prasetyo, 2014).

2.5 Klasifikasi

Klasifikasi adalah proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat

memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Model itu sendiri bisa berupa aturan jika-maka berbentuk pohon pengambilan keputusan (decision tree), formula matematis seperti Bayesian dan Support Vector Machine atau bisa juga berupa jaringan seperti neural network.

Proses klasifikasi biasanya dibagi menjadi 2 fase : *learning* dan *test*. Pada fase learning, sebagian data yang telah diketahui kelas datanya diumpankan untuk membentuk model prediksi. Karena menggunakan data yang telah diberikan label terlebih dulu oleh ahli di bidang itu sebagai contoh data yang benar maka klasifikasi sering juga disebut sebagai metode diawasi (*supervised method*). Kemudian pada fase test-nya model yang sudah terbentuk ditesting dengan sebagian data lainnya untuk mengetahui akurasi dari model tersebut. Bila akurasinya mencukupi model ini dapat dipakai untuk prediksi kelas data yang belum diketahui.



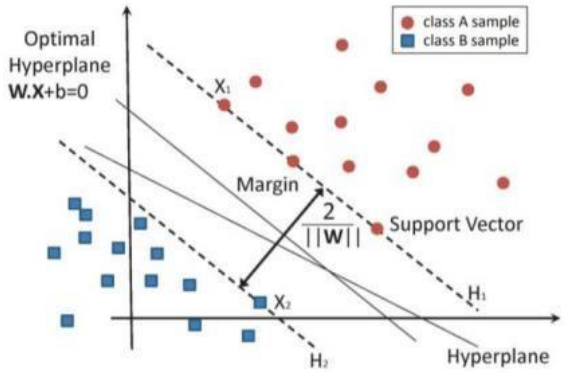
Gambar 2.3 Gambar Skema Klasifikasi

2.6 Algoritma Support Vector Machine (SVM)

Algoritma *Support Vector Machine (SVM)* ditemukan oleh Vladimir N. Vapnik dan Alexey Ya. Chervonenkis pada tahun 1963. Pada tahun 1963. Pada tahun 1992, Bernhard E. Boser, Isabelle M. Guyon dan Vladimir N. Vapnikmengusulkan cara untuk membuat *nonlinier classifier* dengan menerapkan trik

kernel pada *maximum-margin hyperplanes* (Boser, Guyon, & Vapnik, 1992). Kemudian konsep standar SVM saat ini diajukan Corinna Cortes dan Vapnik pada tahun 1993 dan diterbitkan pada tahun 1995 (Cortes & Vapnik, 1995).

Prinsip yang mendasar dari SVM adalah bagaimana mencari fungsi *hyperplane* (garis pemisah) yang dapat memisahkan antara kedua kelas secara maksimal. Maksimal yang dimaksud adalah *hyperplane* dapat memisahkan data kedua kelas dengan margin yang paling baik. Margin merupakan jarak garis *hyperplane* dengan anggota – anggota terdekat dari kedua kelas. Margin yang mampu memisahkan kelas secara maksimal disebut Optimal *Hyperplane*.



Gambar 2.4 Gambar Klasifikasi data menggunakan *Support Vector Machine* (SVM)

Pada gambar 2.4 menunjukkan bahwa H1, H2, dan *hyperplane* merupakan pemisah kedua kelas. Dimana X adalah *dot product* dari variabel dan konstanta pada setiap dan W adalah nilai yang tegak lurus dengan X.

$$w.Xi + b \leq -1 \quad \ldots \quad (1)$$

Persamaan 1 merupakan *hyperplane* yang bersinggungan terhadap data yang ada pada kelas A (H1).

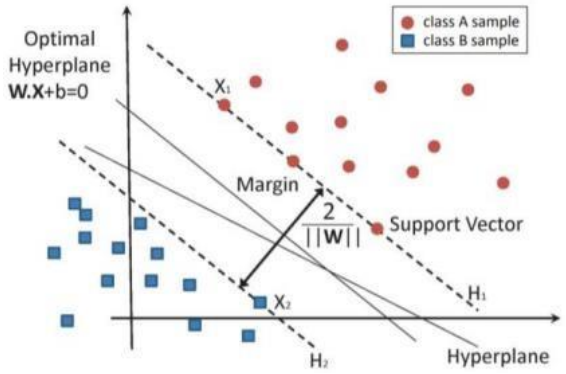
$$w.Xi + b \geq +1 \quad \ldots \quad (2)$$

Persamaan 2 merupakan *hyperplane* yang bersinggungan terhadap data yang ada pada kelas B (H2).

$$w.X + b = 0 \quad \ldots \quad (3)$$

Dan persamaan 3 merupakan *hyperplane* yang berada diantara *hyperplane* kelas A dan kelas B (Garis *hyperplane*). Sedangkan untuk data yang bersinggungan dengan H1 di kelas A dan H3 di kelas B disebut dengan *Support Vector*.

2.6.1 The linierly separable case



Gambar 2.5 Gambar *Linear Hyperplane*

Pertama, pertimbangkan kasus ketika data dapat dipisahkan secara *linear*. Masalah klasifikasi dapat dirumuskan kembali sebagai salah satu menemukan *hyperplane f* (w, b) = $x_i \cdot w + b$ yang memisahkan sampel positif dan negatif:

$$x_i \cdot w + b \geq 1 \text{ untuk } y_i = 1 \quad \ldots \quad (4)$$

$$x_i \cdot w + b \leq 1 \text{ untuk } y_i = -1 \quad \ldots \quad (5)\text{ini}$$

setara dengan persamaan berikut

$$y_i(x_i \cdot w + b) - 1 \geq 0 \text{ untuk } i = 1, \ldots, l \quad (6)$$

keterangan :

- x_i = vektor / titik data ke-i
- y_i = target / kelas data ke-i

- w = vektor weight (bobot) yang menunjukkan tingkat kemiringan garis model
- b = nilai bias yang menunjukkan posisi garis model terhadap sumbu y

$a \cdot b \equiv \sum_i a_i b_i$ menunjukkan *dot product*. Semua titik yang ada pada persamaan (4) terletak pada *hyperplane* H_1 **:** $x_i \cdot w + b = 1$ dengan w normal dan jarak tegak lurus dari titik asal *hyperplane* H_1 yaitu $d_{H1} = \frac{(1-b)}{\|w\|}$. Demikian pula, semua titik yang ada pada persamaan 5 terletak pada *hyperplane* H_2 **:** $x_i \cdot w + b = -1$ dengan jarak dari titik asal *hyperplane* H_2 yaitu $d_{H2} = \frac{(-1-b)}{\|w\|}$.

Jarak antara dua *hyperplane* H_1 dan H_2 disebut sebagai margin dan memiliki persamaan:

$$Margin = | d_{H1} - d_{H2} | = \frac{2}{\|w\|} \quad \ldots \quad (7$$

Keterangan:

- d_{H1} = jarak dari *hyperplane* positif ke *hyperplane* utama
- d_{H2} = jarak dari *hyperplane* negatif ke *hyperplane* utama
- $||w||$ = nilai *magnitude* (besaran) vektor bobot yang menunjukkan panjang garis *hyperplane* utama

Secara umum, ada banyak *hyperplane* yang memenuhi kondisi yang diberikan dalam persamaan (6) untuk kumpulan data yang dapat dipisahkan. Untuk memilih salah satu *hyperplane* dengan kemampuan generalisasi terbaik, maka perlu memaksimalkan *margin* diantara *hyperplane* kanonik (H_1 dan H_2) (lihat gambar 4) (Burges, 1998). Hal ini dibenarkan dari perspektif teori komputasi karena memaksimalkan *margin* sesuai dengan meminimalkan dimensi Vapnik-Chervonenkis (V-C) dari pengklasifikasi yang dihasilkan

(Vapnik, 1997).

Oleh karena itu, permasalahan optimasi yang harus diselesaikan menjadi seperti berikut:

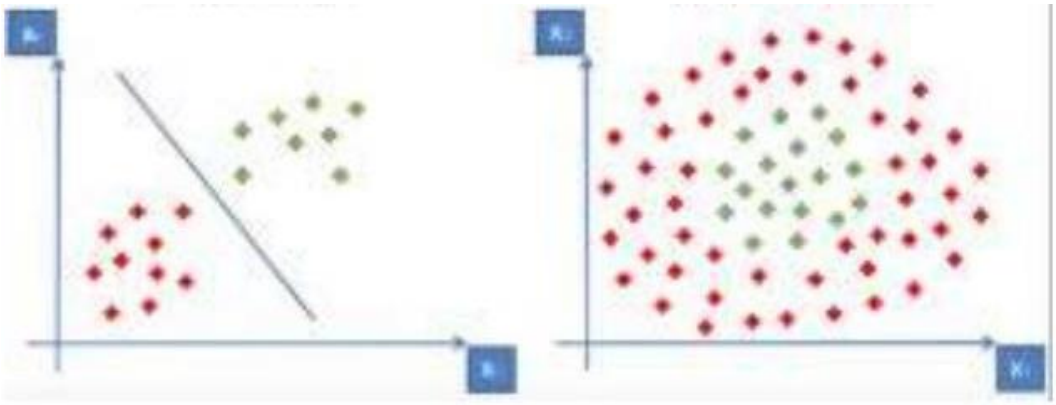
$$\textit{Minimalkan} \qquad J_1[w] = \frac{1}{2}\|w\|^2 \quad \ldots \ (8)$$

$$\textit{dimana} \qquad y_i(x_i \cdot w + b) - 1 \geq 0; \, i \ \ldots \ (9)$$

Keterangan:

- $J_1[w]$ = fungsi optimasi untuk mencari *hyperplane* utama terbaik pada kasus *linearly separable*
- $\|w\|$ = nilai *magnitude* (besaran) vektor bobot yang menunjukkan panjang garis *hyperplane* utama
- x_i = vektor / titik data ke-i
- y_i = target / kelas data ke-i
- w = vektor weight (bobot) yang menunjukkan tingkat kemiringan garis model
- b = nilai bias yang menunjukkan posisi garis model terhadap sumbu y.

2.6.2 The non-linierly separable case



Gambar 2.6 Gambar *Linier* vs. *Non-linearly Separable*

Jika data tidak dapat dipisahkan secara linier, seperti dalam banyak kasus, maka diperkenalkan variabel *slack* $\xi_i \geq 0$ untuk menangani kesalahan klasifikasi sehingga *constraint* berikut terpenuhi:

$$y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0 \text{ untuk } i = 1, \ldots, l \quad \ldots \quad (10)$$

Disini, perlu memilih *tradeoff* antara memaksimalkan margin dan mengurangi jumlah kesalahan klasifikasi dengan menggunakan parameter C yang ditentukan pengguna sehingga permasalahan optimasi menjadi seperti berikut:

$$\textit{Minimalkan} \quad J_2[w, \xi_i] = \left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i\right) \quad \ldots \quad (11)$$

$$\textit{dimana} \quad y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0;$$

$$\xi_i \geq 0;\ i = 1, \ldots, l \quad \ldots \quad (12)$$

Keterangan:

- $J_2[w, \xi_i]$ = fungsi optimasi untuk mencari *hyperplane* utama terbaik pada kasus *non-linearly separable*
- $\|w\|$ = nilai *magnitude* (besaran) vektor bobot yang menunjukkan panjang garis *hyperplane* utama

- C = nilai parameter yang menentukan *tradeoff* antara memaksimalkan margin dan mengurangi jumlah kesalahan klasifikasi
- ξ_i = nilai untuk menangani kesalahan klasifikasi

Permasalahan *constrained optimization* tersebut dapat diubah menjadi permasalahan *unconstrained quadratic convex programming*, dengan memperkenalkan *Lagrangian multipliers* sebagai berikut

$$\begin{aligned} & L(w, b, \xi_i, h, r) \\ & = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \\ Minimalkan \quad & -\sum_{i=1}^{l} h_i[y_i(x_i \cdot w + b) \\ & - 1 + \xi_i] - \sum_{i=1}^{l} r_i\xi_i \end{aligned} \quad \ldots (13)$$

Keterangan:

- $L(w, b, \xi_i, h, r)$ = fungsi optimasi untuk mencari *hyperplane* utama terbaik menggunakan *quadratic programming*
- h_i = nilai *non-negative Lagrange multipliers*
- r_i = nilai *non-negative Lagrange multipliers*

2.6.3 The dual problem and sparsity of solution

Permasalahan optimasi dari persamaan (13) dapat diubah menjadi bentuk *dual problem* yang jauh lebih mudah untuk ditangani dengan melihat titik-titik sadel dari persamaan (13) (Burges, 1998).

$$Maksimalkan \quad L_D(h) = \left(\sum_{i=1}^{l} h_i - \frac{1}{2} h \cdot Dh\right) \quad \ldots \quad (14)$$

$$dimana \quad \sum_{i=1}^{l} y_i h_i = 0; \quad \ldots \quad (15)$$

$$0 \le h_i \le C;\ i = 1, \ldots, l \quad \ldots \quad (16)$$

Keterangan:

- $L_D(h)$ = fungsi optimasi untuk mencari *hyperplane* utama terbaik menggunakan *dual problem*
- $D_{ij} = y_i y_j x_i \cdot x_j$

Solusi untuk *dual problem* ini secara tradisional ditemukan dengan menggunakan *standart quadratic programming packages*. Setelah *optimal multipliers* h_i ditemukan, maka pengklasifikasi menjadi seperti berikut

$$f(x) = sign\left(\sum_{i} h_i y_i x_i \cdot x + b_0\right) \quad \ldots \quad (17)$$

Keterangan:

- $f(x)$ = fungsi *classifier* untuk mengklasifikasikan data uji
- h_i = nilai *Lagrange multipliers*
- y_i = target / kelas data latih ke-i
- x_i = vektor / titik data latih ke-i
- x = vektor / titik data uji
- b_0 = nilai bias yang menunjukkan posisi garis model terhadap sumbu ordinat

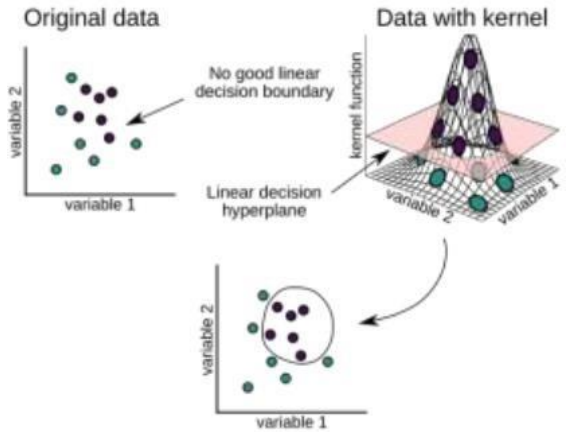
Berdasarkan kondisi optimal *Kuhn-Tucker*, jumlah *non-zero Lagrange multipliers* h_i jauh lebih kecil daripada jumlah data, karena jenis perluasan untuk pengklasifikasi yang diberikan oleh persamaan (17) sangat jarang. Titik data yang sesuai dengan *non-zero Lagrangians* ini disebut sebagai *support vector*. Nilai bias b_0 dapat ditemukan menggunakan *support vectors* x_{sv} apapun

$$b_0 = y_{sv} - \sum_{i \in SV} h_i y_i x_i \cdot x_{sv} \quad \ldots \quad (18)$$

Keterangan:

- b_o = nilai bias yang menunjukkan posisi garis model terhadap sumbu ordinat
- h_i = nilai *Lagrange multipliers*
- y_i = target / kelas data latih ke-i
- x_i = vektor / titik data latih ke-i
- x_{sv} = vektor / titik data latih yang menjadi *support vectors*
- y_{sv} = target / kelas dari data latih yang menjadi *support vectors*.

2.6.4 Extension to non-linier decision surfaces



Gambar 2.7 Gambar Implementasi Kernel

Kekuatan *linear decision surfaces* sangat terbatas. SVM menyediakan cara yang mudah untuk memperluas analisis dari ruang *input* ke ruang fitur nonlinier dengan menggunakan pemetaan dimensi tinggi Φ (x). Menemukan *hyperplane* pemisah linier dalam ruang fitur ini setara dengan menemukan *non-linear decision boundary* di ruang *input*.

Perlu dicatat bahwa vektor *input* hanya muncul dalam bentuk *dot product* $\Phi(x_i) \cdot \Phi(x_j)$ dalam *dual problem* dan solusinya. Dengan menggunakan *kernel* yang mereproduksi ruang Hilbert, maka *dot product* dari dua vektor fitur ini dapat ditulis ulang menjadi *kernel* asalkan beberapa kondisi terpenuhi

$$\begin{aligned}\left(\Phi(x_i) \cdot \Phi(x_j)\right) &= K(x_i, x_j) \\ &\equiv \sum_k \phi_k(x_i)\phi_k(x_j)\end{aligned} \quad \ldots \quad (19)$$

Keterangan:

- x_i = vektor / titik data latih ke-i
- x_j = vektor / titik data latih / uji ke-j
- $\left(\Phi(x_i) \cdot \Phi(x_j)\right) = K(x_i, x_j)$ = fungsi *kernel* yang memetakan data dari ruang input (berdimensi rendah) ke ruang Hilbert (berdimensi tinggi)

Beberapa contoh umum dari *kernel* adalah *Gaussian RBF* dan *polynomial of degree d*

$$\textit{Gaussian RBF:} \quad K(x, x') \equiv e^{-\|x-x'\|^2/2\sigma^2} \quad \ldots \quad (20)$$

$$\textit{Polynomial of degree d:} \quad K(x, x') \equiv (x \cdot x' + 1)^d \quad \ldots \quad (21)$$

Keterangan:

- $K(x, x')$ = fungsi *kernel* yang memetakan data dari ruang input (berdimensi rendah) ke ruang Hilbert (berdimensi tinggi)
- x = vektor / titik data latih
- x' = vektor / titik data latih / uji
- $e \approx 2{,}718$ = *konstanta Euler*
- $\|x - x'\|$ = *vector norm* / jarak antar vektor data
- σ = nilai parameter sigma yang nilainya diatur bebas oleh pengguna
- d = nilai parameter *degree* yang nilainya diatur bebas oleh pengguna

Pengguna *kernel sebagai pengganti dot product* dalam masalah optimasi menyediakan implementasi otomatis untuk merepresentasikan *hyperplane* di ruang fitur, bukan di ruang input (lihat Gambar 2.7).

2.6.5 Mendapatkan *balanced classifier*

Dalam kasus ketika data dapat dipisahkan secara linier, *balanced classifier* didefinisikan sebagai *hyperplane* (f (w, b) = $x \cdot w + b$) yang terletak di tengah dua kelas, atau lebih tepatnya, variabel bias b didefinisikan sebagai berikut:

$$b = -\frac{1}{2}(x_i^+ \cdot w + x_i^- \cdot w) \quad \ldots \quad (22)$$

x_i^+ *dan* x_i^- adalah dua *support vector* dari dua kelas. Tentu saja, *balanced classifier* lebih unggul daripada *unbalanced classifier*. Untuk permasalahan asli (lihat persamaan (8)), *balanced classifier* dapat diperoleh secara otomatis. Namun, saat memaksimalkan margin yang dimodifikasi, *balanced classifier* tidak bisa diperoleh secara otomatis. Batasan pada besaran *w* harus dipenuhi.

Misalkan ada *unbalanced classifier* f (w', b'). Tanpa kehilangan *generality*, maka asumsinya adalah *classifier* akan lebih dekat ke kelas negatif, yaitu:

$$x_i^+ \cdot w' + b' = c;\ c > 1$$

$$x_i^- \cdot w' + b' = -1$$

Hal tersebut mudah untuk diperiksa dengan menggunakan transformasi $w^* = \frac{2}{1+c} w'$ dan $b^* = \frac{2}{1+c} b' + \frac{1-c}{1+c}$ bahwa bisa didapatkan *balanced classifer f* (w^*, b^*) yang sejajar dengan *unbalanced classifier*

$$x_i^+ \cdot w^* + b^* = 1$$

$$x_i^- \cdot w' + b' = -1$$

Jadi, dapat dibangun *parallel classifier* yang diparameterisasi oleh c, untuk c ≥ 1 dengan cara berikut:

$$w(c) = \frac{1+c}{2} w^* \quad \ldots \quad (23)$$

$$b(c) = \frac{1+c}{2} b^* - \frac{1-c}{2} \quad \ldots \quad (24)$$

$$x_i^+ \cdot w(c) + b(c) = c;\ c > 1 \quad \ldots \quad (25)$$

$$x_i^- \cdot w(c) + b(c) = -1 \quad \ldots \quad (26)$$

Disini, f (w(1), b(1)) memberikan *balanced classifier*, yang sesuai dengan c = 1 dan f (w^*, b^*). Untuk c > 1, diperoleh *unbalanced classifier* yang lebih dekat ke kelas negatif. Demi kesederhanaan, maka dapat diasumsikan *augmenting factor* menjadi 1. Kemudian, SVMseq meminimalkan $\|w\|^2 + b^2$. Jadi saat

$$L(c) = \|w(c)\|^2 + b(c)^2 \quad \dots \quad (27)$$

mengambil nilai minimum pada c = 1, SVMseq akan menyediakan *balanced classifier*. Untuk menghitung turunan dari $L(c)$ terhadap c, dapat menggunakan persamaan berikut

$$\frac{dL(c)}{dc} = \frac{1+c}{2}\left[\|w^*\|^2 - \frac{1}{(1+c)^2}\right] + \frac{1+c}{2}\left[b^* + \frac{c}{1+c}\right]^2 \quad \dots \quad (28)$$

Perhatikan bahwa $c \geq 1$ dan $\|\boldsymbol{w}^*\|^2 > \frac{1}{4}$ memastikan $\frac{dL(c)}{dc} > 0$, untuk setiap c.

Jadi, $\|\boldsymbol{w}^*\|^2 > \frac{1}{4}$ adalah kondisi yang cukup untuk memperoleh *balanced classifier* dalam metode ini. Dalam praktiknya, hal ini mudah dipenuhi hanya dengan menskalakan data *input* ke nilai yang lebih kecil, otomatis hasilnya mengarah ke nilai w yang lebih besar.

Bekerja di sepanjang garis pada gambar 6, dan menyelesaikan permasalahan *hyperplane* di ruang fitur dimensi tinggi, maka permasalahan optimasi yang dimodifikasi dapat diubah menjadi bentuk *dual problem*:

$$Maksimalkan \quad L'_D(h) = \left(\sum_{i=1}^{l} h_i - \frac{1}{2} h \cdot D'h\right) \quad \ldots \quad (29)$$

$$dimana \quad 0 \leq h_i \leq C;\ i = 1, \ldots, l \quad \ldots \quad (30)$$

$$D'_{ij} = y_i y_j K(x_i, x_j) + \lambda^2 y_i y_j \quad \ldots \quad (31)$$

Disini, dalam kasus *non-linear*, perlu menambah vektor *input* dengan dimensi ekstra dalam ruang fitur, yaitu, $\Phi(x') = (\Phi(x)\lambda)^T$. *Non-linear classifier* yang dihasilkan dihitung menggunakan persamaan berikut

$$f(x) = sign\left(\sum_{i \in SV} h_i y_i K(x_i, x) + h_i y_i \lambda^2\right) \quad \ldots \quad (31)$$

Keterangan:

- $f(x)$ = fungsi *classifier* untuk mengklasifikasi data uji
- h_i = nilai *Lagrange multipliers*
- y_i = target / kelas data latih ke-i
- x_i = vektor / titik data latih ke-i
- x = vektor / titik data uji
- $K(x_i, x)$ = fungsi kernel yang memetakan data dari ruang input (berdimensi rendah) ke ruang Hilbert (berdimensi tinggi)
- λ = nilai *augmenting factor* sebagai pengganti nilai bias.

2.6.6 Sequential Learning Algorithm

Berikut adalah Algoritma *Sequential Learning SVM* (SVMseq) yang diusulkan oleh (Vijayakumar & Wu, 1999):

1. Inisialisasi $h_i = 0$. Hitung matriks $D_{ij} = y_i y_j \left(K(x_i, x_j) + \lambda^2\right)$ untuk $i, j = 1, \ldots, l$

2. Pada setiap pola, dari $i = 1$ sampai dengan l, hitung

- $E_i = \sum_{j=1}^{l} h_j D_{ij}$
- $\delta h_i = min\{max[\gamma(1 - E_i), -h_i], C - h_i\}$
- $h_i = h_i + \delta h_i$

3. Jika hasil pelatihan sudah *converged*, maka hentikan pelatihan, jika belum maka ulangi langkah kedua.

2.7 Radial Basis Function (RBF) Kernel

RBF kernel merupakan fungsi kernel yang biasa digunakan dalam analisis ketika data tidak terpisah secara linear. RBF kernel memiliki dua parameter yaitu *Gamma* dan *Cost*. Parameter *Cost* atau biasa disebut sebagai C merupakan parameter yang bekerja sebagai pengoptimalan SVM untuk menghindari misklasifikasi di setiap sampel dalam *training dataset*. Parameter *gamma* menentukan seberapa jauh pengaruh dari satu sampel *training dataset* dengan nilai rendah berarti "jauh", dan nilai tinggi berarti "dekat". Dengan *gamma* yang rendah, titik yang berada jauh dari garis pemisah yang masuk akal dipertimbangkan dalam perhitungan untuk garis pemisah. Ketika *gamma* tinggi berarti titik-titik berada di sekitar garis yang masuk akal akan dalam perhitungan (Patel, 2017).

2.8 Akurasi

Tingkat akurasi sebuah *classifier* untuk data *testing* adalah rasio perbandingan jumlah data testing yang dapat diklasifikasikan dengan benar dengan jumlah seluruh data. Akurasi dapat dinyatakan dalam satuan persen dengan mengalikan hasil perhitungan dengan 100%.

2.9 Python

Python merupakan salah satu bahasa pemrograman komputer yang populer dan sering digunakan terutama dalam melakukan perhitungan. Bahasapemrograman ini dapat dijalankan hamper di semua platform, seperti *Linux*, *Windows*, dan *Mac*. Pada bahasa pemrograman Python, deklarasi suatu variabel dapat dilakukan secara langsung tanpa menyebutkan tipe datanya.

Bahasa pemrograman Python diciptakan oleh Guido Van Rossum, dan diperkenalkan pertama kali pada Centrum Wiskunde & Informatica (CWI) di Belanda pada awal tahun 1990-an. Bahasa pemrograman yang terinspirasi dari bahasa pemrograman ABC ini pertama kali dikembangkan pada awal tahun 1990 oleh Guido Van Rossum di Stiching Mathematisch Centrum. Pada tahun 1995, pengembangan bahasa pemrograman Python dilakukan di Corporation forNational Research Initiatives. Saat ini *Python Software Foundation* bertugas sebagai pengembang *Python*.

2.10 Anaconda

Anaconda dikembangkan oleh Organisasi Anaconda, Inc. Anacondamerupakan sebuah perangkat lunak (software) yang bisa digunakan untuk bahasa pemrograman Python dan R, serta berisi beberapa paket tambahan gunapemrograman data science, komputasi ilmiah (scientific computing) seperti machine learning, data processing skala luas, analisis prediksi, matematika, hinggateknik dalam satu distribusi yang user friendly. Anaconda menyediakan banyak pustaka atau library yang sudah diinstall sebelumnya. Beberapa di antaranya adalah NumPy, SciPy, Pandas, Scikit learn, nltk, dan Jupiter.

2.11 Penelitian Terdahulu

Penelitian terdahulu ini menjadi salah satu acuan untuk mendapatkan uraian teori, temuan, dan bahan penelitian lainnya yang digunakan sebagai landasan penyusunan kerangka penelitian dari perumusan masalah yang diteliti. Adapun pembahasan temuan penelitian terdahulu adalah sebagai berikut:

2.11.1 Penelitian I

Tabel 2.11.1 review singkat penelitian terdahulu I

Judul Penelitian	Tujuan Penelitian	Metode Penelitian	Hasil	Perbedaan dengan Penelitian
Deteksi Dini Penyakit Diabetes Menggunakan Machine Learning dengan Algoritma Logistic Regression	Mendapatkan prediksi penyakit diabetes dengan akurasi yang terbaik	Metode yang digunakan dalam penelitian tersebut menggunakan Algoritma *Logistic Regression*	Dari hasil yang dilakukan didapatkan nilai akurasi sebesar kisaran 75 - 80%	Pada penelitian yang akan dilakukan saat ini menggunakan algoritma yang berbeda yakni Algoritma SVM

2.11.2 Penelitian II

Tabel 2.11.2 review singkat penelitian terdahulu II

Judul Penelitian	Tujuan Penelitian	Metode Penelitian	Hasil	Perbedaan dengan Penelitian
Perbandingan Algoritma Naïve Bayes dan K-Nearest Neighbor Pada Imbalace Class Dataset Penyakit Diabetes	Mengklasifikasi penyakit diabetes dengan metode machine learning	Metode yang digunakan menggunakan algoritma Naïve Bayes dan KNN	Dari hasil yang dilakukan didapatkan nilai akurasi sebesar kisaran 71 - 83%	Pada penelitian yang akan dilakukan saat ini menggunakan algoritma yang berbeda yakni SVM

2.11.3 Penelitian III

Tabel 2.11.3 review singkat penelitian terdahulu III

Judul Penelitian	Tujuan Penelitian	Metode Penelitian	Hasil	Perbedaan dengan Penelitian
Sistem Prediksi Penyakit Diabetes Berbasis Decision Tree	Memprediksi penyakit diabetes dengan menggunakan metode Decision Tree	Metode yang digunakan pada penelitian tersebut ialah algoritma Decision Tree	Dari hasil yang dilakukan didapatkan nilai akurasi sebesar 84%	Pada penelitian yang akan dilakukan saat ini menggunakan algoritma yang berbeda yakni SVM