BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian yang sudah dilakukan sebagai solusi untuk menyelesaikan masalah data *imbalanced* dengan menggunakan beberapa metode serta teknik optimasi sehingga memperoleh performa *accuracy* yang tepat dan akurat. Salah satu metode yang mampu menyelesaikan masalah data *imbalanced* yaitu *cost sensitive* yang di implementasikan pada metode *decision tree*. Metode tersebut bekerja dengan meminimalkan kesalahan *decision tree*. Penelitian yang telah dilakukan dengan menggunakan metode tersebut antara lain:

Tabel 2.1 Penelitian Neelesh Haldankar Akash

Author, Year	Neelesh Haldankar Akash, 2016
Problem	Memprediksi resiko pengajuan nasabah baru untuk menghindari pinjaman macet pada bank
Scope of problem	Data penelitian diperoleh di UCI *Machine Leaming Repository* dengan menggunakan 1000 data, 20 atribut dan 2 class *imbalanced*.
Method	• Fitur seleksi *atribut* menggunakan metode *boruta* • Algoritma *decision tree* menggunakan *random forest* • *Cost sensitive* menerapkan metode *thresholding* untuk menghitung minimum *cost.*
Result	Accuracy yang diperoleh sebesar 76%

Tabel 2.2 Penelitian Atefeh Daraei

Author, Year	Atefeh Daraei, 2016
Problem	Memprediksi pasien yang terkena penyakit *Myocardial infarction* (MI) dengan jumlah data *imbalanced*.
Scope of problem	Data penelitian diambil dari rumah sakit di Shahid Madani Specialized**,** Iran. Data yang diperoleh sebanyak 750 data dengan 295 pasien didiagnosa mengidap penyakit MI dan 455 di diagnosa sehat. Terdiri dari 92 atribut dan 2 class.
Method	• Fitur seleksi *atribut* menggunakan metode *genetic algoritma* • Algoritma *decision tree* menggunakan J4.5 • *Cost sensitive* menggunakan algoritma *metacost* untuk menghitung minimum *cost*.
Result	Hasil accuracy terbaik diperoleh dengan menggunakan fitur seleksi GA sebesar 82.67%.

Tabel 2.3 Penelitian Neelesh Haldankar Akash

Author, Year	Hong Zhao, et al, 2015
Problem	Melakukan pengujian klasifikasi *dataset*, tanpa adanya *preprocessing* / seleksi fitur terlebih dahulu untuk menghasilkan data baru yang akan diproses.
Scope of problem	Dataset yang digunakan diperoleh dari *UCI Machine Leaming Repository* sebanyak 8 dataset yaitu: No. | Name | Domain | $\|U\|$ | $\|C\|$ | d 1 | *Biodeg* | Chemical | 1055 | 41 | Type 2 | *Breast* | Clinic | 699 | 9 | Class 3 | *Clean* | Physics | 476 | 166 | Class 4 | *Credit* | Commerce | 1000 | 20 | Class 5 | *Diabetes* | Clinic | 768 | 8 | Class 6 | *Eeg* | Life | 1923 | 14 | Class 7 | *Wdbc* | Clinic | 569 | 30 | Diagnosis 8 | *Wilt* | Clinic | 4839 | 5 | Class
Method	• Algoritma *decision tree* menggunakan C4.5 • *Cost sensitive* menggunakan teknik *probabilistic pruning mechanism* untuk menghitung minimum *cost.*
Result	• Metode *Cost sensitive decision tree* berhasil menurunkan *cost* dibawah 10%.

Tabel 2.4 Penelitian Shichao Zhang

Author, Year	Shichao Zhang, 2005
Problem	*Dataset* yang terdapat *missing value* dapat menjadi permasalahan karena jika dihapus dapat menggurangi kandungan informasi dari data.
Scope of problem	Dataset yang digunakan diperoleh dari *UCI Machine Leaming Repository* sebanyak 5 dataset imbalanced yaitu: | | No. of Attributes | No. of Examples | Class distribution (N/P) | | Ecoli | 6 | 332 | 230/102 | | Breast | 9 | 683 | 444/239 | | Heart | 8 | 161 | 98/163 | | Thyroid | 24 | 2000 | 1762/238 | | Australia | 15 | 653 | 296/357 |
Method	Menggunakan metode *cost sensitive* dengan menggunakan 4 *strategy misclassified cos*t yaitu: *The Null Strategy, The Intemal Node Strategy, The C4.5 Strategy*
Result	Hasil dari 4 strategy yang diujikan *intemal node strategy* menjadi metode terbaik dalam menyelesaikan permasalahan data *missing value.*

Tabel 2.5 Penelitian Polat

Author, Year	Polat dan Gunes, 2009
Problem	Menyelesaikan masalah klasifikasi *imbalanced multiclass*
Scope of problem	*dataset* dari UCI *machine leaming repository*, diantaranya *dataset Dermatology*, *Image Segmentation*, dan *Lymphography*.
Method	• Allgoritma pengklasifikasi C4.5 dengan pendekatan *One-Against-All* • Pengujian menggunakan teknik 10-*fold cross validation*
Result	Hasil accuracy yang lebih tinggi yaitu 96,71%, 95,18%, dan 87,95% pada masing-masing *dataset*.

Tabel 2.6 Penelitian Aitkenhead

Author, Year	Aitkenhead, 2008
Problem	Mengatasi masalah data *noise* dan masalah kombinasi data kuantitatif dan kualitatif pada *dataset* yang dapat menyulitkan proses kategorisasi kelas
Scope of problem	Dataset yang digunakan adalah *Glass Chemistry* dan *Car Costing*
Method	mengembangkan pendekatan *evolusioner* pada algoritma *Decision Tree C4.5*
Result	Hasil accuracy yang lebih tinggi yaitu 82,40% dan 89,20% pada masing-masing *dataset*.

Tabel 2.7 Penelitian Balamurugan

Author, Year	Balamurugan dan Rajaram, 2009
Problem	Mengatasi masalah kegagalan dalam pemilihan *atribut* yang akan di-*split* yang dapat menyebabkan label kelas dipilih secara acak.
Scope of problem	*Dataset* yang digunakan dalam penelitian tersebut diantaranya *Blood Transfusion*, *Teaching Assistant Evaluation*, *SPECT Heart*, *Haberman's Survival*, *Contraceptive Method Choice*, *Hayes Roth*, *Concrete*, *Forest-fires*, *Solarflare 1*, dan *Solarflare 2*.
Method	Algoritma *Decision Tree C4.5* dengan menambahkan prosedur penghitungan *Maximum Influence Factor* (MIF)
Result	Hasil penelitian tersebut didapatkan nilai accuracy lebih tinggi yaitu 85,16 %, 77,78 %, 71,70 %, 78,79 %, 77,50 %, 76,74 %, 76,74 %, 75,68 %, 77,09% pada masing-masing *dataset*.

Tabel 2.8 Penelitian Xiaoyong Chai

Author, Year	Xiaoyong Chai, Lin Deng and Qiang Yang, 2004
Problem	Data *imbalanced* terdapat *Missing value* pada *atribut* dapat menurunkan performa *classifier.*
Scope of problem	*Dataset* yang digunakan diperoleh dari *UCI Machine Leaming Repository* sebanyak 8 *dataset imbalanced* yaitu:
Method	*Cost sensitive naïve bayes*. Menggunakan metode klasifikasi *naïve bayes* dan *cost sensitive* digunakan untuk mencari jumlah minimal *cost* dari jumlah *cost* rata-rata *misclassification cost* dan *test cost.*
Result	*Cost sensitive naïve bayes* memiliki nilai total *cost* rata-rata *misclassification cost* dan *test cost* lebih baik dibandingkan dengan *cost sensitive decision tree.*

Name of datasets	No. of attributes	Name of datasets	No. of attributes
Ecoli	6	Breast	9
Heart	8	Thyroid	24
Australia	15	Cars	6
Voting	16	Mushroom	22

Tabel 2.9 Penelitian Khor Kok-Chin

Author, Year	Khor Kok-Chin dan Ng Keng-Hoong, 2016
Problem	Data *imbalanced dua class* untuk menyeleksi customer bank yang ingin menggunakan produk bank
Scope of problem	*Dataset* yang digunakan diperoleh dari *UCI Machine Leaming Repository*. Data dikumpulkan 45.211 dari 14 organisasi marketing oleh bank Portugal. Data memiliki atribut sebanyak 16 dan 2 class dan memiliki sifat data numeric dan nominal
Method	Penelitian ini menggunakan 3 metode pengujian yaitu *cost sensitive naïve bayes, cost sensitive decision tree* C4.5 dan *cost sensitive naïve bayes tree*.
Result	*Cost sensitive naïve bayes* memiliki nilai nilai *accuracy* terbaik sebesar 96.30%, *Cost sensitive decision tree C4.5* memiliki nilai nilai *accuracy* terbaik sebesar 96.50% dan *Cost sensitive naïve bayes tree* memiliki nilai nilai *accuracy* terbaik sebesar 96.70%,

Pada tabel 2.1, 2.2 dan 2.3 merupakan penelitian yang menggunakan seleksi atribut dan metode *cost sensitive decision tree C4.5*. Hasil dari metode tersebut mampu meningkatkan performa *accuracy* pada data *imbalanced*. Pada tabel 2.4, 2.5, 2.6 dan 2.7 penelitian tersebut

menggabungkan teknik optimasi dengan metode *cost sensitive decision tree* C4.5. Teknik tersebut mampu meningkatkan performa *accuracy classifier*. Tabel 2.8 menggunakan metode *cost sensitive* pada *naïve bayes* dan *decision tree C4.5* hasilnya *cost sensitive naïve bayes* memilki performa yang lebih baik dari pada menggunakan *decision tree C4.5.*

2.2 *Data Mining*

Data mining merupakan bidang keahlian yang menggabungkan teknik dari *machine leaming* (mesin pembelajaran), *pattem recognition* (pengenalan pola), analisis, statistik dan visualisasi sebagai solusi permasalahan untuk mendapatkan informasi dari *database.* Pola-pola informasi tersebut dilakukan analisa untuk dipelajari sebagai perilaku yang terstuktur guna memberikan wawasan dan pengetahuan untuk menggambil keputusan (Larose, 2005).

Data mining merupakan bagian dari *knowledge discovery in database* (KDD). Sebelum mengambil keputusan, langkah awal dilakukan beberapa tahapan proses antara lain pengumpulan data sebagai masukan, pemrosesan data, hingga keluaran berupa informasi. Secara umum tahapan pengolahan data hingga memperoleh informasi digambarkan seperi gambar 2.1.

Gambar 2.1. Skema Proses KDD

Data awal menjadi masukan dari proses KDD yang didalamnya mengandung informasi. Proses pengolahan data awal (*preprocessing*) diperlukan sebelum masuk pada proses *data mining*. Pada proses ini, data diproses menjadi keluaran yang diinginkan dan dibutuhkan dalam proses *data mining*. Proses ini juga menggabungankan data dari berbagai sumber menjadi satu, menghilangkan *noise*, *menghapus* data duplikat, dan pemilihan *atribut* yang relevan.

Pre-processing merupakan proses yang memerlukan banyak sumber daya, karena proses ini merupakan tahapan penting dari keseluruhan proses KDD. Hal ini disebabkan karena kualitas data masukan memberikan pengaruh terhadap hasil keluaran proses. Tahap *data mining* dilakukan setelah tahap *pre-processing* data. Informasi diperoleh dengan melakukan

observasi pola dan hubungan dalam data menggunakan algoritma *machine leaming*. Tahap terakhir adalah *post-processing*, dimana dalam tahap ini hasil penggalian data akan diintegrasikan dengan sistem pengambilan keputusan atau divisualisasikan untuk memudahkan pengguna dalam menganalisa informasi. *Data mining* secara umum terbagi menjadi dua jenis yaitu sebagai berikut:

1. ***Prediktif.*** Teknik ini bertujuan untuk memprediksi target *class* berdasarkan nilai dari karakteristik *atribut* lainnya. Teknik jenis prediktif antara lain Klasifikasi dan *regresi*.
2. ***Deskriptif.*** Teknik ini bertujuan untuk mendapatkan pola yang menghasilkan simpulan hubungan dari data. Metode ini digunakan untuk menjelaskan pola dan hubungan data. Teknik jenis deskriptif antara lain *Clustering* dan *asosiasi*.

2.3 *Pre-processing*

preprocessing adalah suatu proses yang dilakukan untuk membuat data awal menjadi data yang berkualitas (*input* yang baik untuk *data mining*). Teknik menghasilkan data yang berkualitas melalui beberapa proses tahapan antara lain pembersihan data, pengurangan data dan *transformasi* data. Proses pembersihan data seperti pengisian nilai kosong (*missing value*), menghapus nilai duplikasi, *noise* dan *outlier*. Proses *transformasi* data dengan menggunakan metode *normalisasi* atau *agregasi*.

2.3.1 Normalisasi Min-Max

Normalisasi merupakan suatu proses *transformasi* data. Salah satu teknik yang digunakan dalam proses *normalisasi* yaitu *min-max.* Teknik *min-max* merupakan metode *normalisasi* dengan cara pengskalaan nilai data antara 0 sampai 1. Tujuan teknik tersebut menyeimbangkan data dalam menentukan nilai informasi yang terkandung dalam sebuah data. Persamaan *min-max* ditunjukkan dibawah ini. (Nasution, 2019)

$$X_n = \frac{X_0 - X_{\min}}{X_{\max} - X_{\min}} \qquad \text{..... (2.1)}$$

Keterangan:

X_n = Nilai data *normalisasi* (0 - 1)

X_0 = Nilai data *aktual*

X_{min} = Nilai *minimum* data aktual keseluruhan

X_{max} = Nilai *maksimum* data aktual keseluruhan

2.3.2 Seleksi *Atribut* PSO

Pemilihan *atribut* menjadi faktor penting untuk menghasilkan data yang berkualitas dalam proses *data mining*. Oleh sebab itu, perlu adanya seleksi atribut untuk memilih atribut relevan dan mengahapus atribut yang tidak relevan sehingga menghasilkan pengetahuan yang *informatif* dan akurat. Salah satu metode yang baik untuk menyeleksi *atribut* adalah *particle swarm optimization* (PSO). PSO merupakan sebuah metode optimasi yang dapat digunakan untuk menyeleksi *atribut* berdasarkan nilai bobot informasi (w).

Metode PSO terinspirasi dari perilaku sekawanan hewan seperti ikan atau burung dalam mencari makan. PSO mengansumsikan atribut sebagai sekumpulan partikel dimana sekumpulan partikel mencari posisi dan kecepatan optimal dalam ruang fitur. Kelebihan metode optimasi *Particle Swarm Optimization* memiliki konsep yang sederhana, mudah diimplementasikan, dan efisien dalam perhitungan jika dibandingkan dengan algoritma matematika dan teknik *optimisasi heuristik* lainnya Kemudian proses seleksinya sebagai berikut (Shih Wei, KuoChing, Shih-Chieh, & Zne-Jung, 2008).

1. Mengansumsikan jumlah kawanan partikel adalah *N*. Kecepatan dan posisi awal pada setiap partikel dalam *N* dimensi ditentukan secara *random* (acak).
2. Menghitung kecepatan semua partikel. Semua partikel bergerak menuju titik optimal yaitu 1. kecepatan awal dari semua partikel diasumsikan sama dengan nol.
3. Membandingkan hasil dari nilai *fitness* setiap partikel dengan nilai sebelumnya. Jika nilai *fitness* setiap partikel pada posisi saat ini lebih baik dari Pbest, maka Pbest diatur untuk posisi saat ini.
4. Membandingkan nilai *fitness* partikel dengan Gbest. Jika Gbest yang terbaik maka Gbest yang di *update*.
5. Persamaan (2.1) dan (2.2) ditunjukkan di bawah ini untuk memperbaharui (*update*) kecepatan (*velocity*) dan posisi (*position*) setiap partikel.

$$V_{id}^{k+1} = w \times V_{id}^{k} + c_1 \times rand_1 \times \left(P_{id} - X_{id}\right) + c_2 \times rand_2 \times \left(G_{id} - X_{id}\right) \quad \text{……. (2.2)}$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \quad \text{…… (2.3)}$$

Keterangan:

V_{id} = Komponen kecepatan individu ke *i* pada *d* dimensi
X_{id} = Posisi individu *i* pada *d* dimensi
w = Parameter inertia *weight*
$c_1 c_2$ = Konstanta akselerasi (*leaming rate*), nilainya antara 0 sampai 1
$rand_{1,2}$ = Parameter random antara 0 sampai 1
P_{id} = *Pbest* (*local best*) individu *i* pada *d* dimensi
G_{id} = *Gbest* (*global best*) pada *d* dimensi

2.4 Klasifikasi

Klasifikasi merupakan salah satu teknik dalam *data mining* dan termasuk ke dalam jenis *supervised methods*. *Supervised methods* merupakan pelabelan *class* yang telah didefinisikan berdasarkan nilai karakteristik atribut. Pada *dataset*, label *class* telah didefinisikan sehingga algoritma pembelajar dapat mempelajari pola atau hubungan antar *atribut* dengan label *class*, sehingga klasifikasi disebut dengan *supervised methods*. Klasifikasi dilakukan dua tahap proses, yaitu proses pembelajaran, dimana proses untuk menemukan pola data dan proses klasifikasi, dimana pola klasifikasi digunakan untuk memprediksi label *class*.

Pada tahap pertama yaitu tahap pembelajaran, algoritma *machine leaming* digunakan untuk menganalisis pola atau hubungan dari *atribut* data dalam menentukan label *class* atau target *atribut*. Pola hasil pembelajaran kemudian dilakukan pengujian untuk dinilai performanya sebagai *classifier*. Jika hasil klasifikasi dinilai memiliki performa yang baik, maka pola klasifikasi dapat diterapkan pada kasus nyata. Klasifikasi memilki beberapa metode antara lain *k*-NN (k-*Nearest Neighbor*), SVM (*support vector machine*), NB (*Naïve Bayes)* dan DT (*decision tree).*

2.4.1 *K-Nearest Neighbor*

Algoritma *K-Nearest Neighbor* (K-NN) merupakan metode klasifikasi objek berdasarkan data pembelajaran jarak yang paling dekat dengan objek tersebut. Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing – masing dimensi merepresentasikan fitur dari data.

Algoritma KNN merupakan algoritma *supervised* yang bertujuan untuk menemukan pola baru dalam data dengan menghubungkan pola data yang sudah ada dengan data yang baru. Tujuan dari algoritma KNN adalah untuk mengklasifikasi objek baru berdasarkan *atribut* dan *training samples*. Dimana hasil dari *sampel* uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada KNN. Pada proses pengklasifikasian, algoritma ini tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori.

Algoritma KNN mengklasifikasikan berdasarkan ketetanggaan sebagai nilai prediksi dari contoh data uji yang baru. Jarak yang digunakan adalah jarak *Euclidean Distance*. Jarak *Euclidean* adalah jarak yang paling umum digunakan pada data *numeric*. Algoritma KNN merupakan algoritma yang menentukan nilai jarak pada data pengujian (*testing*) dengan data pelatihan (*training*) berdasarkan nilai terkecil dari nilai ketetanggaan terdekat. Algoritma *k*-NN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sampel uji yang baru. Jarak *Euclidean* adalah jarak yang paling umum digunakan pada data numerik. *Euclidean distance* didefinisikan sebagai berikut. (Krisandi, 2013)

$$d(x_i, x_j) = \sqrt{\sum_r^n (a_r(x_i) - a_r(x_j))^2} \quad \text{..... (2.4)}$$

Keterangan :

$d(x_i, x_j)$ = Jarak *Euclidean (Eucledean Distance)*

(x_i) = *Record* ke-i

(x_j) = *Record* ke-j

(a_r) = Data ke-r

i, j = 1,2,3……n

Algoritma k-NN mencari nilai jarak pada data *testing* dengan data *training* berdasarkan nilai terkecil dari nilai ketetangaan terdekat. Didefinisikan sebagai berikut (Krisandi, 2013).

$$D_{nn}(C_1, C_2) = \min_{1 \le i \le r, 1 \le j \le s} d\left(y_i, z_j\right) \quad \text{..... (2.5)}$$

Tabel 2.10 menunjukan mengenai kelebihan dan kekurangan metode K-NN pada saat proses klasifikasi.

Tabel 2.10 Kelebihan dan kekurangan metode K-NN

Kelebihan	Kekurangan
• Mampu mengklasifikasikan data yang bersifat *non-liniear* • Mudah dipahami dan diimplementasikan • *Efektif* meskipun data *training*-nya berjumlah besar • Mampu memprediksi label *class* berdasarkan *instance* paling dekat.	• Menentukan Parameter k (Jumlah Tetangga Terdekat) • Tidak bisa menangani *Missing Value* • *Sensitif* terhadap data *Outlier* • Rentan terhadap atribut yang *non-Informatif* • Rentan terhadap jumlah data yang besar • Waktu komputasi yang lama

Sumber: Mutrofin, 2010

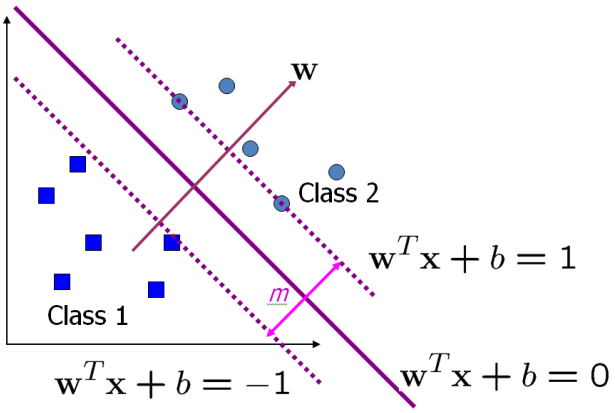
2.4.2 *Support Vector Machine*

Support Vector Machine (SVM) adalah suatu metode prediksi yang menggunakan teknik klasifikasi maupun *regresi* (Santosa, 2007). SVM memiliki prinsip dasar *linier classifier* yaitu mengklasifikasikan data dengan cara memisahkan secara *linier*, namun SVM telah dikembangkan agar dapat bekerja pada *problem non-linier* dengan memasukkan konsep *kemel* pada ruang kerja berdimensi tinggi.

SVM memisahkan data dengan cara membangun sebuah bidang pemisah (*hyperplane)*. *Hyperplane* yang baik tidak hanya mampu digunakan untuk memisahkan data, akan tetapi juga memiliki batasan (*margin*) yang besar. Pencarian *hyperplane* inilah yang menjadi inti dari *support vector machine*. Namun, dalam proses pencarian *hyperplane* tersebut, akan muncul permasalahan baru yaitu sebuah formula yang sangat sulit untuk dipecahkan disebut dengan permasalahan *Quadratic Programming*.

Konsep SVM sebagai solusi untuk mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah *class* yaitu + dan - pada input *space* (*discrimination boundaries*). *Margin* adalah jarak antara *hyperplane* dengan pola terdekat dari masing-masing *class*. Pola yang paling dekat ini disebut sebagai *support vector*. Mencari posisi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM. Pada ruang berdimensi tinggi, akan dicari *hyperplane* yang dapat memaksimalkan jarak (*margin*) antara *class* data. *Hyperplane* klasifikasi *linier* SVM dinotasikan sebagai berikut (Santosa, 2007).

$$f(x) = w^T x + b \qquad (2.6)$$



Gambar 2.2 Model klasifikasi SVM

Tabel 2.11 menunjukan mengenai kelebihan dan kekurangan metode SVM pada saat proses klasifikasi.

Tabel 2.11 Kelebihan dan kekurangan metode SVM

Kelebihan	Kekurangan
• Mampu mengklasifikasikan data yang tidak masuk dalam *class* atau kategori • Mampu menghadapi masalah pada suatu pola (*Curse of dimensionality).* • SVM dapat diimplementasikan dengan mudah.	• SVM sangat sulit jika menggunakan data dengan jumlah yang besar • SVM perlu ditambahkan teknik pengolahan lanjut supaya mampu menyelasaikan masalah pada data *multiclass.*

Sumber: Nugroho, 2003

2.4.3 *Naive Bayes*

Naive Bayes merupakan sebuah metode klasifikasi berdasarkan perhitungan dari sekumpulan probabilitas dengan menjumlahkan frekuensi dan kombinasi nilai dari *dataset* yang diberikan. Algoritma menggunakan teorema *Bayes* dan mengasumsikan bahwa semua *atribut* tidak saling ketergantungan (*independen*). Karena diasumsikan sebagai variable *independent*, maka hanya varians dari suatu variabel dalam sebuah *class* yang dibutuhkan untuk menentukan klasifikasi, bukan keseluruhan dari *matriks kovarians*.

Kelebihan menggunakan *Naive Bayes* adalah bahwa metode ini hanya membutuhkan jumlah data pelatihan (*training data*) yang sedikit untuk menentukan *estimasi* paremeter yang diperlukan dalam proses pengklasifikasian. Berikut ini persamaan umum dari *Naïve Bay*es (Indranandita, 2008).

$$P(Y = y \mid X_1, X_2, ..., X_p) = \frac{P(y)P(X_1, X_2, ..., X_P \mid y)}{P(X_1, X_2, ..., X_P)} \quad \text{..... (2.7)}$$

Dengan asumsi *independent* $X_1, X_2, ..., X_P$ didapat $P(X_1, X_2, ..., X_P \mid y) = P(X_1 \mid y)P(X_2 \mid y)P(X_3 \mid y).....P(X_p \mid y)$ sehingga probabilitas masing-masing *class* dalam klasifikasi *naïve bayes* adalah

$$P(y \mid X_1, X_2, ..., X_p) = \frac{P(y)P(X_1 \mid y)P(X_2 \mid y).....P(X_p \mid y)}{P(X_1, X_2, ..., X_P)} = \frac{P(y)\prod_{i=1}^{p} P(X_i \mid y)}{P(X_1, X_2, ..., X_P)} \quad \text{... .. (2.8)}$$

Keterangan:

X = Data dengan *class* yang belum diketahui

Y = Hipotesis data merupakan suatu *class* spesifik

$P(Y|X)$ = Probabilitas hipotesis Y berdasar kondisi X (*posteriori probabilitas*)

$P(Y)$ = Probabilitas hipotesis Y (*prior probabilitas*)

$P(X|Y)$ = Probabilitas X berdasarkan kondisi pada hipotesis Y

$P(X)$ = Probabilitas X

Tabel 2.12 menunjukan mengenai kelebihan dan kekurangan metode *Naïve Bayes* pada saat proses klasifikasi.

Tabel 2.12 Kelebihan dan kekurangan metode *Naïve Bayes*

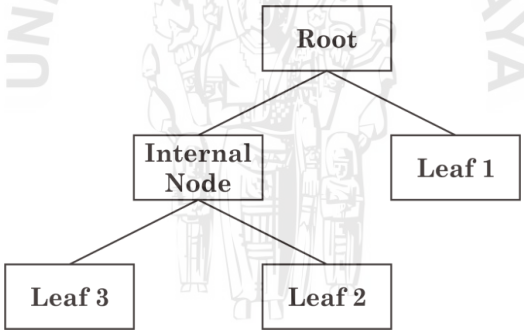
Kelebihan	Kekurangan
• Mudah untuk dipahami • Waktu komputasi Lebih cepat • Mampu menangani data *kuantitatif* dan data *diskrit* • Memerlukan jumlah data pelatihan yang sedikit • Mampu menangani *missing value* • Efisiensi pada ruang memori penyimpanan • Mampu menyelesaikan masalah data *imbalanced*	• Jumlah data training yang sedikit menyebabkan kurang akurat untuk membuktikan kebenaran prediksi klasifikasi • Tidak berlaku jika probabilitas kondisionalnya adalah 0 (nol), apabila nol maka probabilitas prediksi akan bemilai nol juga • Mengasumsikan variabel bebas

Sumber: Manalu, 2017

2.4.4 *Decision Tree*

Decision tree atau pohon keputusan merupakan salah satu metode klasifikasi yang paling terkenal karena mudah untuk dipahami dan diterapkan oleh manusia dan komputer. Proses dari *decision tree* adalah mengubah bentuk data (*tabel*) menjadi pola model pohon, mengubah pola model pohon menjadi aturan (*rule*), dan menyederhanakan *rule*. Manfaat utama dari penggunaan *decision tree* adalah kemampuannya untuk mem*break down* proses pengambilan keputusan yang kompleks menjadi lebih sederhana sehingga lebih mudah dalam menganalisa pola keputusan. *Decision tree* juga berguna untuk mengobservasi data, menemukan hubungan tersembunyi antara variabel *input* dengan sebuah variabel target.

Pola model *decision tree* dibentuk oleh *node-node* antara lain *root node*, *intemal node* dan *leaf node*. *Root node* merupakan *node* yang berada pada posisi puncak yang menghasilkan cabang kriteria. Cabang kriteria disebut juga dengan *intemal node* yang berasal dari *node* atasnya dan memiliki cabang kebawah. *Leaf node* merupakan target digunakan dalam menggambil keputusan. Konsep dari pohon keputusan ditampilkan pada gambar dibawah ini.



Gambar 2.3. Konsep dasar pohon keputusan

Pada *decision tree* untuk menentukan *node* sebagai pola model pohon keputusan dibutuhkan sebuah algortima. Algoritma yang biasanya digunakan antara lain ID3, C4.5 dan C5.0. Tabel 2.13 merupakan perbedaan antara algoritma ID3, C4.5 dan C5.0 seperti ditunjukan seperti table di bawah ini.

Tabel 2.13 Perbandingan algoritma ID3, C4.5 dan C5.0

	ID3	C4.5	C5.0
Type of data	Kategori	Kontinue dan kategori	Kontinue dan kategori, *Dates, Times, Timestamps*
Speed	Lambat	Lebih cepat dari ID3	Paling cepat dari ID3 dan C4.5
Pruning	*No*	*Pre-pruning*	*Pre-pruning*
Boosting	Tidak *support*	Tidak *support*	*Support*
Missing Value	Tidak dapat menyelesaikan	Tidak dapat menyelesaikan	Dapat menyelesaikan
Formula	*information entropy*	*split info and gain ratio*	*information Gain*

Sumber: Kumar, 2015

Tabel 2.14 menunjukan mengenai kelebihan dan kekurangan metode *Decision Tree* pada saat proses klasifikasi.

Tabel 2.14 Kelebihan dan kekurangan metode *Decision Tree*

Kelebihan	Kekurangan
• Waktu komputasi yang cepat • Pengambilan keputusan lebih simple dan spesifik • Efisien dalam merancang model • Fleksibel sehingga akan meningkatkan kualitas keputusan yang dihasilkan • Dapat mengolah data numerik • Model yang dihasilkan mudah dipahami • Dapat menyelesaikan masalah data *imbalanced*	• Terjadi *overlaping* • Pengakumulasian jumlah kesalahan dari setiap tingkat dalam sebuah pohon keputusan yang besar • Kesulitan dalam mendesain pohon keputusan yang optimal • Hasil kualitas keputusan sangat tergantung pada bagaimana pohon tersebut didesain. • Kebutuhan ruang penyimpanan yang besar

Sumber: Astuti, 2019

2.4.4.1 Algoritma ID3

Algoritma Pohon Keputusan ID3 (*Iterative Dichotomiser* 3) merupakan sebuah metode yang digunakan untuk membuat pola model pohon keputusan. Algoritma ini menggunakan konsep dari *entropy informasi* untuk menentukan setiap node. Algoritma ini melakukan pencarian secara menyeluruh pada semua kemungkinan pohon keputusan. Secara ringkas, langkah kerja Algoritma ID3 dapat digambarkan sebagai berikut:

1. Hitung *Entropy (H)* dan *Information gain (G)* dari setiap atribut dengan menggunakan rumus sebagai berikut. (Hssina, 2013)

$$Entropy(S) = -\sum_{i=1}^{n} p(s_i)\log_2(p(s_i)) \quad \text{..... (2.9)}$$

$$Gain(S, A_i) = Entropy(S) - \sum_{\alpha \in A_i}^{n} \frac{|S_\alpha|}{|S|} Entropy(S_\alpha) \quad \text{..... (2.10)}$$

Keterangan:

S = Himpunan kasus
A = Atribut
n = Jumlah partisi atribut A
$|S_\alpha|$ = Jumlah kasus pada partisi ke-i
$|S|$ = Jumlah kasus dalam S
p_i = Proporsi dari S_i terhadap S

2. Bentuk simpul yang berisi atribut tersebut.
3. Proses terus dilakukan perhitungan information gain sampai seluruh data telah masuk dalam kelas yang sama. Atribut yang telah dipilih tidak diikutkan lagi dalam perhitungan nilai information gain.

2.4.4.2 Algoritma C4.5

Algoritma pohon keputusan C4.5 adalah pengembangan dari algoritma ID3. Sehingga algoritma C4.5 memiliki prinsip dasar kerja yang sama dengan algoritma ID3. Perbedaan utama C4.5 dari ID3 adalah:

1. C4.5 mampu menangani atribut kontinyu dan diskrit.
2. C4.5 mampu menangani *training* data dengan *missing value*.
3. Hasil pohon keputusan C4.5 akan di*pruning* setelah dibentuk.
4. Pemilihan *node* yang dilakukan dengan menggunakan *Gain Ratio*.

Algoritma C4.5 menggunakan *gain ratio* untuk memperbaiki *information gain*, dengan rumus *gain ratio* (Merbah, 2014).

$$GainRatio(S, A) = \frac{Gain(S, A)}{Split\,\text{inf}\,o(S, A)} \quad \text{..... (2.11)}$$

Atribut dengan nilai *Gain Ratio* tertinggi dipilih sebagai *node root* untuk simpul. Dengan *gain* adalah *information gain*. Pendekatan ini menerapkan normalisasi pada *information gain* dengan menggunakan apa yang disebut sebagai *split information*. *SplitInfo* menyatakan *entropy* atau informasi potensial dengan rumus:

$$Split\inf o(S,A) = -\sum_{i=1}^{n} \frac{|S_\alpha|}{|S|} \log_2 \frac{|S_\alpha|}{|S|} \quad \text{..... (2.12)}$$

Keterangan:

S = Himpunan kasus

A = Atribut

n = Jumlah partisi atribut A

$|S_\alpha|$ = Jumlah kasus pada partisi ke-i

$|S|$ = Jumlah kasus dalam S

2.4.4.3 Algoritma C5.0

Algoritma C5.0 menjadi penyempura dari algoritma sebelumnya yaitu C4.5 dan ID3. Dibandingkan dengan C4.5, algoritma C5.0 lebih cepat dalam mengambil keputusan dan lebih efektif dalam membangun struktur pohon keputusan. *Information gain* atau biasa disebut *gain info* adalah kriteria pemisahan yang menggunakan pengukuran *entropy*. Untuk memperoleh *information gain* dari suatu atribut dibutuhkan *entropy* keseluruhan *class* atau *Entropy* (S). *Entropy* (S) adalah banyaknya jumlah *bit* yang dibutuhkan untuk mengekstrak suatu *class* dari sekumpulan data acak pada ruang sample S. Semakin kecil nilai *entropy* maka memiliki hasil yang baik untuk digunakan dalam mengekstraksi suatu *class*. Nilai informasi yang dinyatakan sebagai panjang kode adalah p2log− *bits* yang memiliki probabilitas p. *Entropy* merupakan suatu parameter yang digunakan untuk mengukur heterogenitas (keberagaman) dari suatu kumpulan data sampel. Semakin besar nilai entropy maka jumlah kriteria pada data sampel semakin heterogen. Secara matematis *entropy* dirumuskan sebagai berikut (Han et al, 2011).

$$H(S) = \sum_{i=1}^{n} -p(s_i)\log_2(p(s_i)) \quad \text{..... (2.13)}$$

Setelah mendapatkan nilai *entropy* selanjutnya mencari nilai *information gain. Information gain* digunakan untuk menggukur efektivitas karakteristik *atribut* dalam mengklasifikasikan *class*. Persamaan 2.14 digunakan untuk menghitung *information gain* sebagai berikut:

$$IG(S, A_i) = H(S) - \sum_{\alpha \in A_i}^{n} \frac{|S_\alpha|}{|S|} H(S_\alpha) \quad \text{……..…….. (2.14)}$$

Keterangan:

H = *Entropy*

S = Himpunan kasus

A = *Atribut*

n = Jumlah partisi atribut A

$|S_\alpha|$ = Jumlah kasus pada partisi ke-i

$|S|$ = Jumlah kasus dalam S

p_i = Proporsi dari S_i terhadap S

Dengan kata lain, *Gain* (A) adalah jumlah keseluruhan nilai kriteria pada atribut A terhadap jumlah total data sampel. Atribut dengan nilai gain terbesar dipilih sebagai simpul atribut (*node root*). Sedangkan cabang (*node intemal*) dibuat berdasarkan pada kriteria pada node sebelumnya hingga menghasilkan keputusan (*leaf node*).

2.5 Metacost

Cost sensitive leaming adalah sebuah metode pembelajaran yang dapat menyelesaikan permasalahan data *imbalanced.* Konsep metode ini adalah menggurangi *cost* dari *missclassification* sehingga dapat meningkatkan performa pada *classifier*. Metode ini mengansumsikan bahwa minoritas *class* mempunyai *cost* lebih besar dari pada mayoritas *class* sehingga *machine leaming* akan fokus mengkalsifikasikan minoritas *class* dengan akurat. *Output* dari *cost sensitive leaming* berupa nilai *cost function. Cost function* merupakan bobot yang diberikan karena kesalahan klasifikasi.

Cost fuction dinotasikan sebagai berikut *C*(i, j) dimana *missclasification* terjadi ketika aktual *class* i diprediksi menjadi *class* j. *Cost sensitive decision tree* dapat diintegrasikan

dalam empat cara. Pertama, *cost function* digunakan sebagai kriteria untuk pemilihan atribut yang akan displit. Kedua, *cost function* digunakan untuk memilih bagian tree yang akan dipangkas (*pruning*). Ketiga, *cost function* digunakan memanipulasi bobot dari data *training* sehingga menghasilkan tree dengan *cost* minimal. Keempat, *cost function* digunakan sebagai kriteria untuk pelabelan *leaf node*.

Cost sensitive leaming dapat dikategorikan menjadi dua kriteria. Kriteria pertama yaitu *direct method*, metode ini antara lain ICET dan *cost sensitive decision tree.* Kriteria kedua yaitu *Meta Leaming*, dibagi menjadi dua *thresholding* dan *sampling.* Untuk struktur dari *cost sensitive leaming* dapat dilihat pada tabel 2.15.

Tabel 2.15 Struktur Model *Cost Sensitive Decision Tree*

Cost Sensitive Leaming	
Direct Method	**Meta Leaming**
1. ICET	**Thresholding**
2. Cost Sensitive Decision Tree	1. Metacost
	2. Cost Sensitive Classifier
	3. Cost Sensitive Naïve Bayes
	4. Emperical Thresholding
	Sampling
	1. Coasting
	2. Weighting

Sumber: Ling, 2008

Metacost merupakan metode *cost sensitive leaming* menggunakan teknik *thresholding meta leaming*. Metode ini dapat diimplementasikan pada *classifier decision tree.* Prinsip kerja dari metode ini adalah menghitung *probability* setiap *leaf node.* Ketika nilai *probability* pada *leaf node* > 1, ada dua cara dalam mengevaluasi model yaitu *pruning* dan *relabeling* untuk mendapatkan model dengan *minimum cost*. *Pruning* merupakan teknik pemangkasan cabang pohon sedangkan teknik *relabeling* merupakan pemberian *label class* baru pada *leaf node* berdasarkan jumlah *class* paling banyak (Domingos, 1999).

Algoritma : Metacost
Output : Decision Tree
Step :

For i = 1 to m
Let S_i be a leaf node of S with n node.
Let M_i = Model produced by applying L to S_i.
For each node x in S
For each class j

Let $P(j|x) = \frac{1}{\sum_i 1} \sum_i P(j|x, M_i)$

if $P(j|x, M_i) = 0$ is produced by M_i

Else $P(j|x, M_i) > 1$

Let x's class = $\text{argmin}_i \sum_j P(j|x) C(i, j)$
Let M = Model produced by applying L to S.
Retum M

Keterangan:

S = Training set
L = Algoritma classification leaming
C = Cost matrix
m = Jumlah leaf node model
= Jumlah node setiap leaf node

2.6 *Validation*

Cross validation (CV) adalah sebuah metode analisa yang dapat digunakan untuk mengevaluasi kinerja dari *classifer* dengan cara, *dataset* dibagi menjadi dua *subset* yaitu data pembelajaran dan data pengujian. Selanjutnya pemilihan jenis CV berdasarkan pada besamya ukuran *dataset*. *K-fold* CV merupakan metode yang baik digunakan karena dapat meminimalkan waktu komputasi dengan tetap mempertimbangkan keakuratan estimasi.

Cara kerja dari *k-fold cross validation* adalah membagi *dataset* menjadi dua kelompok yaitu data *traning* dan data *testing*, kemudian dilakukan proses *testing* yang dilakukan pengulangan sebanyak *k* kali. Hasil *testing* itu kemudian dirata-rata untuk menghasilkan sebuah nilai *accuracy*. Nilai *k* yang terlalu kecil dapat mempengaruhi nilai *accuracy* menjadi

rendah. Hal tersebut disebabkan karena nilai *k* yang kecil, mudah terpengaruh oleh *noise*. Sedangkan, nilai *k* yang terlalu besar menghasilkan proses yang kurang efektif. Hal ini disebabkan nilai *k* yang besar, membutuhkan waktu yang cukup lama dalam melakukan pengujian. Metode 10 *fold cross validation* menjadi metode standar untuk pembelajaran dan pengujian data (Catal & Diri, 2009). Berikut contoh ilustrasi 10-*fold cross validation.*

Tabel 2.16 Ilustrasi 10 *fold cross validation*

Validasi ke-n	**Data Set**									
	1	2	3	4	5	6	7	8	9	10
1	**test**	train	train	train	train	train	train	train	train	train
2	train	**test**	train	train	train	train	train	train	train	train
3	train	train	**test**	train	train	train	train	train	train	train
4	train	train	train	**test**	train	train	train	train	train	train
5	train	train	train	train	**test**	train	train	train	train	train
6	train	train	train	train	train	**test**	train	train	train	train
7	train	train	train	train	train	train	**test**	train	train	train
8	train	train	train	train	train	train	train	**test**	train	train
9	train	train	train	train	train	train	train	train	**test**	train
10	train	train	train	train	train	train	train	train	train	**test**

Sumber: Catal & Diri, 2009

Berdasarkan Tabel 2.16 menjelaskan bahwa nilai *fold* yang digunakan 10-*fold cross validation*. Berikut adalah langkah-langkah pengujian data menggunakan 10-*fold cross validation.*

a. Membagi *dataset* menjadi 10 bagian, yaitu D_1,D_2,D_3,D_4,…….,D_{10} t= (1,2,3,….,10) digunakan sebagai data *training* dan *dataset* lainnya sebagai data *testing.*
b. Nilai *accuracy* dihitung pada setiap *iterasi* (*iterasi*-1, *iterasi*-2*,….., iterasi*-10) kemudian dihitung rata-rata nilai *accuracy* dari seluruh iterasi untuk mendapatkan nilai *accuracy* dari data keseluruhan.

2.7 Evaluasi Performansi Model Klasifikasi

Pengukuran terhadap kinerja suatu sistem klasifikasi merupakan hal yang penting. Kinerja sistem klasifikasi menggambarkan seberapa baik sistem dalam mengklasifikasikan

data. Dalam tahap ini ada dua tahap pengukuran performansi pengujian model klasifikasi *cost sensitive decision tree* yaitu dengan menggunakan metode *confution matrix* dan *cost matrix.*

2.7.1 *Confution Matrix Multi-class*

Confusion matrix merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja *classifier* dengan baik. Pada dasamya *confusion matrix* mengandung informasi dengan membandingkan hasil prediksi yang dilakukan oleh sistem dengan nilai aktual yang sebenamya. Berikut merupakan gambaran hasil evaluasi klasifikasi *multi-class* seperti ditampilkan pada Tabel 2.17 berikut:

Tabel 2.17 *confusion matrix multi-class*

Class Actual	*Class Prediction*					Total
	1	2	3	…	K	
1	x_{11}	x_{12}	x_{13}	…	x_{1k}	n_1
2	x_{21}	x_{22}	x_{23}	…	x_{2k}	n_2
3	x_{31}	x_{32}	x_{33}	…	x_{3k}	n_3
…	…	…	…	…	…	…
k	x_{k1}	x_{k2}	x_{k3}	…	x_{kk}	n_k
Total	n_1	n_2	n_3	…	n_k	N_{total}

Sumber : (Akbar, Yudhistira dan Cholissodin, 2014)

$$TP = X_{11} + X_{22} + X_{33} + ... + X_{KK}$$
$$TN = (X_{11} + X_{22}) + (X_{11} + X_{33}) + (X_{22} + X_{33}) + (X_{11} + X_{KK}) + (X_{22} + X_{KK}) + (X_{33} + X_{KK})$$
$$FP = (X_{21} + X_{31} + ... + X_{K1}) + (X_{12} + X_{32} + ... + X_{K2}) + (X_{13} + X_{23} + ... + X_{K3}) + ...$$
$$FN = (X_{12} + X_{13} + ... + X_{1K}) + (X_{21} + X_{23} + ... + X_{2K}) + (X_{31} + X_{32} + ... + X_{3K}) + ...$$

$$n_1 = X_{11} + X_{12} + X_{13} + ... + X_{1K}$$
$$n_2 = X_{21} + X_{22} + X_{23} + ... + X_{2K}$$
$$n_3 = X_{31} + X_{32} + X_{33} + ... + X_{3K}$$
$$n_k = X_{K1} + X_{K2} + X_{K3} + ... + X_{KK}$$
$$N_{total} = n_1 + n_2 + n_3 + ... + n_K$$

1. *True Postive* (TP) menunjukan bahwa *class* yang diprediksi positif sedangkan *class* aktualnya adalah positif.
2. *True Negatif* (TN) menunjukan bahwa *class* yang diprediksi negatif sedangkan *class* aktualnya adalah negatif.
3. *False Positif* (FP) menunjukan bahwa *class* yang diprediksi negatif sedangkan *class* aktualnya adalah positif.
4. *False Negatif* (FN) menunjukan bahwa *class* yang diprediksi positif sedangkan *class* aktualnya adalah negatif.

Pengukuran performa dari data *imbalanced* dapat dilakukan dengan menggunakan perhitungan nilai *recall, precision* dan *f-measure*. *Accuracy* klasifikasi menunjukkan performansi model klasifikasi secara keseluruhan, dimana semakin tinggi *accuracy* maka tingkat ketepatan antara nilai prediksi mendekati dengan nilai aktual (Powers, 2011).

$$Accuracy = \frac{TP}{N_{total}} \qquad \text{(2.15)}$$

Recall digunakan untuk mengukur tingkat positif (*minor*) benar atau mengukur tingkat keberhasilan sistem dalam menemukan lagi informasi pada data acak. *Precision* adalah tingkat negative (*mayor*) benar atau tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Persamaa *recall* dan *Precision* adalah sebagai berikut (Powers, 2011).

$$\text{Re}call = \frac{TN}{TN + FP} \qquad \text{(2.16)}$$

$$\text{Pr}ecision = \frac{TP}{(TP + FP)} \times 100\% \qquad \text{(2.17)}$$

F-measure merupakan kombinasi dari nilai *recall* dan *precision* yang digunakan untuk Mengukur performasi prediksi pada *classier* berdasarkan gabungan hasil nilai *precision* dan *recall*. Dengan menggunakan persamaan seperti berikut (Cao dkk dalam Sain, 2013).

$$F - Mesure = 2 \times \frac{sensitivity \times specificity}{sensitivity + specificity} \qquad \text{(2.18)}$$

2.7.2 *Cost Matrix Multi-class*

Cost matrix adalah representasi dari hasil kesalahan prediksi (*misclassification*) dalam *machine leaming. Missclassification cost* pada *multiclass* dinotasikan sebagai berikut C_{12}, C_{13}, C_{21}, C_{23}, C_{31}, C_{32}, C_{kk}. Ketika klasifikasi benar dinotasika C_{11}, C_{22}, C_{33}, C_{kk} nilai *cost* adalah 0. *Cost matrix* ditampilakan Tabel 2.18.

Tabel 2.18 *cost matrix multi-class*

Class Actual	Class Prediction					Total
	1	2	3	…	K	
1	C_{11}	C_{12}	C_{13}	…	C_{1k}	n_1
2	C_{21}	C_{22}	C_{23}	…	C_{2k}	n_2
3	C_{31}	C_{32}	C_{33}	…	C_{3k}	n_3
…	…	…	…	…	…	…
k	C_{k1}	C_{k2}	C_{k3}	…	C_{kk}	n_k
Total	n_1	n_2	n_3	…	n_k	N_{total}

Total *cost* menunjukkan performansi model klasifikasi secara keseluruhan berdasarkan hasil *misclassification cost*, dimana semakin kecil nilai *cost* klasifikasi . Hal ini berarti semakin baik performansi model klasifikasi. Persamaan untuk menghitung total *cost* ditunjukan seperti berikut (George, 2017).

$$Cost = N_{total}\left[(C_{FN} + FN) * (C_{FP} + FP)\right] \quad \text{(2.19)}$$