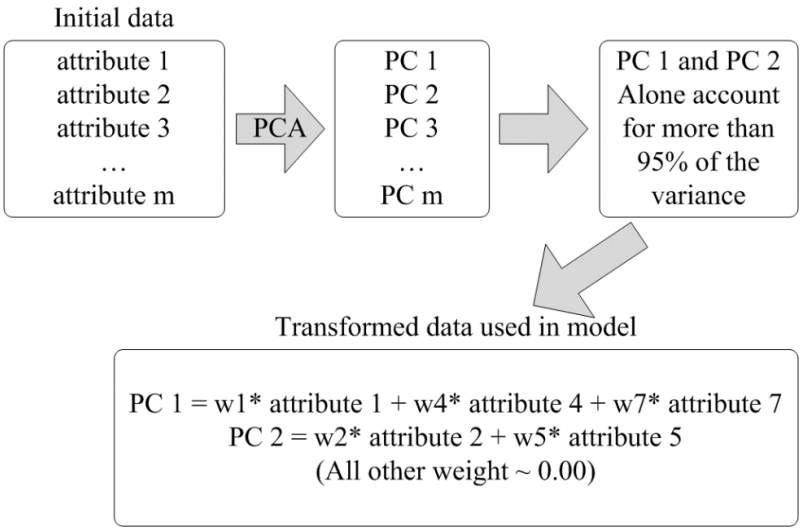
BAB II

TINJAUAN PUSTAKA

II.1 *Principal Component Analysis* (PCA)

PCA merupakan kombinasi linear dari variabel awal yang secara geometris kombinasi linear ini merupakan sistem koordinat baru yang diperoleh dari rotasi sistem semula. Metoda PCA sangat berguna digunakan jika data yang ada memiliki jumlah variabel yang besar dan memiliki korelasi antar variabelnya. Perhitungan dari *principal component analysis* didasarkan pada perhitungan nilai *eigen* dan vektor *eigen* yang menyatakan penyebaran data dari suatu *dataset*. Dengan menggunakan PCA, variabel yang tadinya sebanyak *n* variabel akan diseleksi menjadi *k* variabel baru yang disebut *principal component*, dengan jumlah *k* lebih sedikit dari *n*. Dengan hanya menggunakan *k principal component* akan menghasilkan nilai yang sama dengan menggunakan *n* variabel. Variabel hasil dari seleksi disebut *principal component*.

PCA digunakan untuk menjelaskan struktur matriks varians-kovarians dari suatu set variabel melalui kombinasi linier dari variabel-variabel tersebut. Secara umum *principal component* (PC) dapat berguna untuk seleksi fitur dan interpretasi variabelvariabel. Skema konseptual yang diilustrasikan bagaimana PCA dapat membantu untuk menyederhanakan dimensi dari data melalui hipotesa *dataset* berjumlah *m* variabel dapat ditunjukkan pada Gambar II.1 berikut:



Gambar II.1 Model konseptual PCA untuk tahap seleksi fitur

Sumber: (Kotu & Deshpande, 2015)

Prosedur pengerjaan *Principal Component Analysis* bertujuan untuk menyederhanakan dan menghilangkan faktor atau indikator yang kurang dominan dan kurang relevan tanpa mengurangi maksud dan tujuan dari data asli dari variabel acak *x* (matrik berukuran *n* x *n*, dimana baris-baris yang berisi observasi sebanyak *n* dari variabel acak *x*) adalah sebagai berikut:

1. Menghitung Matrik Varians Kovarian dari Data Observasi. *Varians (Var(x))* dihitung untuk menemukan penyebaran data dalam *dataset* untuk menentukan penyimpangan data dalam set data sampel. Matrik Kovarian *Cov(x,y)* ialah matriks yang nilai-nilai kovariansi pada tiap cell-nya diperoleh dari sampel. Misalkan *x* dan *y* adalah variabel acak.

$$Var\ (x) = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(\tilde{z}_{ij} - \mu_j)^2 \quad (2.1)$$

$$Cov\ (x, y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \mu_{xj})\ (y_{ij} - \mu_{yj}) \quad (2.2)$$

Dengan μx dan μy merupakan rata – rata *(mean)* sampel dari variabel *x* dan *y*, dimana x_i dan y_i merupakan nilai observasi ke-*i* dari variabel *x* dan *y*. Dari data nilai yang digunakan, maka diperoleh matrik kovarian berukuran *n* x *n*.

2. Mencari *Eigen values* dan *Eigen vector* dari Matrik Kovarian. Nilai *eigen* dan vektor *eigen* untuk matriks kovarians dihitung. Nilai *eigen* yang dikomputasi kemudian ditransformasikan (rotasi *orthogonal varimax*) menggunakan persamaan berikut:

$$Det\ (A - \lambda I) = 0 \tag{2.3}$$

dimana:

A = matrik *n* x *n*

λ = nilai *eigen value*

I = matriks identitas (matriks persegi dengan elemen diagonal utama bernilai 1 sedangkan elemen lain bernilai 0

3. Menentukan Nilai Proporsi *Principal Component* (Proporsi *Principal Component* (%))

$$PC(\%) = \frac{Nilai\ Eigen}{Varians\ Covarian} \times 100\% \tag{2.4}$$

4. Menghitung Bobot Faktor *(Factor Loading)* Berdasarkan *Eigen vector* dengan persamaan:

$$A_x = \lambda_x \tag{2.5}$$

Sehingga diperoleh kombinasi linear yaitu:

λ1, λ2, λ3… λn adalah *eigen value* matrik A

x1, x2, x3… xn adalah *eigen vector* sesuai *eigen value*-nya (λn)

Persamaan *eigen value* dan *eigen vector* merupakan *Eigen Value Decomposition* (EVD), dengan persamaan sebagai berikut:

$$A\,X = X\,D$$

$$A = X\,D\,X^{-1} \tag{2.6}$$

dimana:

A = matrik *n* x *n* yang memiliki *n eigen value* (λn)

D = *eigen value* dari *eigen vector*-nya

X = *eigen vector* dari matrik *A*

X^{-1}= invers dari *eigen vector X*

II.2 *Singular Value Decomposition* (SVD)

SVD sering digunakan untuk membangun suatu *low rank approximation* pada matriks *term-by-document* dalam retrival infomasi (Albright et al., 2014). Nyatanya, untuk membangun suatu *rank-k approximation* A_k pada matriks *A rank r term-by-document*, gunakan hanya komponen *singular k* yang paling signifikan dimana $k < r$.

$$A_k = \sum_{i=1}^{k} \sigma_i\, u_i v_i^T = U_k \sum_k V_k^T \tag{2.7}$$

Dimana σ_i adalah nilai *singular A* ke-*i*, u_i dan v_i^T adalah vektor *singular*. Teknik menggantikan matriks *A* dengan A_k yang sudah direduksi disebut *Latent Semantic Indexing* (LSI) karena *low rank approximation* mengungkapkan makna dan hubungan antar dokumen yang tersembunyi atau *laten* dalam data matriks asli.

SVD yang terpotong memberikan dasar yang bagus dimana semua *low rank approximation* yang lain dapat dinilai untuk akurasi kuantitif. Matriks *A* selalu

merupakan suatu matriks non negatif, akan tetapi komponen *singular* tandanya (+/) bercampur. Hilangnya struktur non negatif pada matriks berarti bahwa faktor dari SVD tidak memberikan interpretabilitas.

Beberapa keuntungan dari SVD, antara lain:

1. Optimal. SVD yang dipotong menghasilkan *approximation rank – k* yang terbaik.
2. Cepat dan komputasi yang *robust*.
3. Faktorisasi yang unik. Inisialisasi tidak mempengaruhi algoritma SVD.
4. Orthogonalitas, menghasilkan basis vektor yang *orthogonal* dan mampu mengkonseptualisasi data asli sebagai vektor dalam ruang.

II.3 *Information Gain* (IG)

IG merupakan metode seleksi fitur paling sederhana dengan melakukan perangkingan atribut dan banyak digunakan dalam aplikasi kategorisasi teks, analisis data *microarray* dan analisis data citra (Chormunge & Jena, 2016). IG dapat membantu mengurangi *noise* yang disebabkan oleh fitur-fitur yang tidak relevan. IG mendeteksi fitur-fitur yang paling banyak memiliki informasi berdasarkan kelas tertentu. Penentuan atribut terbaik dilakukan dengan menghitung nilai *entropy* terlebih dahulu. *Entropy* merupakan ukuran ketidakpastian dapat digunakan untuk menyimpulkan distribusi fitur dalam bentuk yang ringkas (Bereziński et al., 2015). Untuk menghitung *entropy* di tunjukkan pada persamaan 2.8.

$$Entropy\ (S) = \sum_{i=1}^{n} -pi \times log_2\ pi \tag{2.8}$$

dimana:

S = himpunan kasus

n = jumlah partisi *S*

pi = proporsi *subset* dari *S* pada partisi ke-*i*

Setelah mendapat nilai *entropy*, maka perhitungan nilai IG dapat dilakukan dengan menggunakan persamaan 2.9.

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \times Entropy(S_i) \quad (2.9)$$

dimana:

S = keseluruhan *dataset*

A = atribut *subset*

n = jumlah partisi atribut *A*

| *Si* | = ukuran *subset* dari *dataset* yang dimiliki atribut *A* pada partisi ke-*i*

| *S* | = ukuran jumlah kasus dalam *dataset*

II.4 *Gain Ratio* (GR)

Salah satu pembobotan menggunakan GR, GR dapat memperbaiki data yang tidak stabil, cocok untuk data numerik dua kelas, sederhana sehingga komputasi lebih cepat. Untuk menghitung *gain ratio* diperlukan pemisahan informasi *(split)* (Ezzat Ibrahim et al., 2012). Untuk *split* informasi digunakan persamaan 2.10.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{|D|}\right) \quad (2.10)$$

dimana:

D = keseluruhan *dataset*

A = atribut *subset*

v = jumlah partisi atribut A

$| D_j |$ = ukuran *subset* dari *dataset* yang dimiliki atribut A partisi ke-j

$| D |$ = ukuran jumlah kasus dalam *dataset*

Selanjutnya *gain ratio* dihitung dengan persamaan 2.11.

$$Gain\ Ratio\ (A) = -\frac{Gain\ (A)}{SplitInfo\ (A)} \tag{2.11}$$

II.5 *Chi-Square*

Metode lain untuk pembobotan atribut disebut pembobotan *chi-square* yang digunakan dalam penelitian yang diberikan oleh (Liu & Motoda, 1998) dengan menemukan nilai dengan menghitung *chi-square* jarak antara setiap atribut input dan atribut kelas.

Motivasi untuk menggunakan *chi-square* sebagai ukuran informasi timbal balik dalam tugas pembobotan atribut adalah karena kemampuan ukuran ini dalam menentukan peringkat atribut (Witten & Frank, 2005). *Chi-square* dirancang untuk bekerja untuk data kategorikal dan tidak berfungsi untuk data kuantitatif. Juga harus diketahui bahwa data tidak boleh di bentuk persentase, hanya frekuensi atau jumlah data yang diperlukan.

Salah satu cara untuk menjelaskan *chi-square* adalah bahwa ia memeriksa hipotesis nol yang menyatakan bahwa tidak ada hubungan antara atribut. Berdasarkan hipotesis nol ini, dibuat suatu model yang mendistribusikan data dalam kategori dengan asumsi bahwa tidak ada hubungan antara atribut. Tes ini didasarkan pada membandingkan distribusi aktual data dan distribusi data yang diharapkan.

Frekuensi yang diamati untuk sel c_{ij} adalah n_{ij}. Frekuensi yang diharapkan untuk sel c_{ij} adalah e_{ij}.

$$e_{ij} = \frac{n_i n_j}{n} \tag{2.12}$$

Rumus uji *chi-square* dapat dihitung sebagai berikut:

$$(x^2) = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(n_{ij} - e_{ij})^2}{e_{ij}} \tag{2.13}$$

dimana:

n = frekuensi yang diharapkan

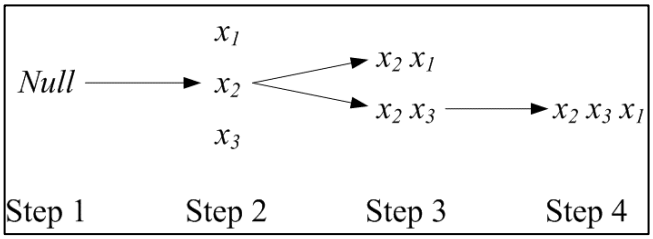
e = frekuensi yang diperoleh

x^2 = nilai *chi-square*

II.6 *Forward Selection*

Pada *forward selection*, dimulai dengan suatu model yang tidak menggunakan fitur apapun sama sekali, yaitu meng-*assign* kelas yang paling sering muncul di *dataset*, pada *input*. Setelah itu, tambahkan satu per satu fitur pada setiap langkah. Langkah berikutnya, kita gunakan satu fitur. Diantara *F* pilihan fitur, cari fitur yang memberi nilai terbaik. Kemudian, pada tahap berikutnya, kombinasikan fitur yang dipilih pada langkah sebelumnya dengan fitur yang tersisa. Hal ini terus diulang sampai menggunakan seluruh fitur pada model.

Untuk mencari model terbaik, hanya perlu mencari kombinasi fitur yang memberikan nilai kinerja terbaik. *Forward selection* diilustrasikan pada Gambar II.2.



Gambar II.2 Ilustrasi *forward selection* untuk tiga fitur

Sumber: (Putra, 2020)

dirumuskan (Alpaydın, 2010):

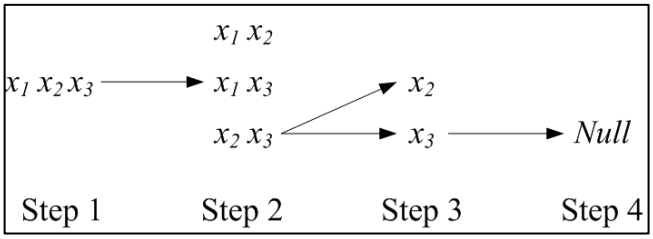
$$j = \arg\min_i E\ (F \cup x_i) \tag{2.14}$$

dan

$$\text{add } x_j \text{ to } F \text{ if } E\ (F \cup x_i) < E(F) \tag{2.15}$$

II.7 *Backward elimination*

Backward elimination adalah kebalikan dari *forward selection*. Apabila pada *forward selection*, kita menambahkan satu fitur tiap langkah, *backward elimination* mengurangi satu fitur pada tiap langkah. Ilustrasi diberikan pada Gambar II.3. Seperti *forward selection*, *backward elimination* juga hanya mencoba sebanyak $F(F + 1)/2$ kombinasi.



Gambar II.3 Ilustrasi *backward elimination* untuk tiga fitur

Sumber: (Putra, 2020)

dirumuskan (Alpaydın, 2010):

$$j = \arg\min_i E\ (F - x_i) \tag{2.16}$$

dan

$$\text{remove } x_j \text{ from } F \text{ if } E\ (F - x_i) < E(F) \tag{2.17}$$

II.8 *Decision Tree*

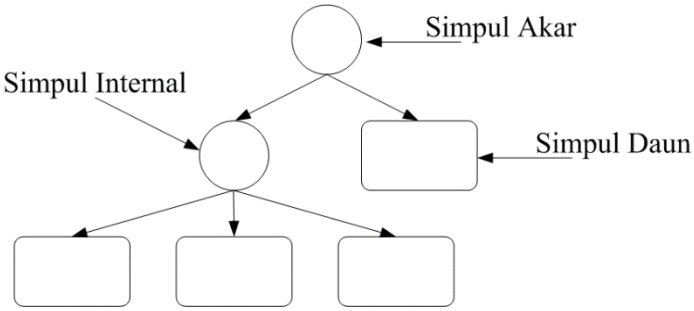
Decision tree merupakan struktur data yang bersifat hirarkikal yang diimplementasikan dari *divide-and-conquer strategy* dan *efficient nonparametric method* serta dapat digunakan pada metode klasifikasi dan metode regresi (Alpaydın, 2010). *Decision tree* merupakan algoritma *machine learning* dalam teknik klasifikasi yang bersifat *supervised learning* untuk membentuk pohon keputusan dari data. Pohon yang terbentuk menyerupai pohon terbalik, dimana akar *(root)* berada di bagian paling atas dan daun *(leaf)* berada di bagian paling bawah (Abellán & Castellano, 2017).

Data dalam pohon keputusan biasanya dinyatakan dalam bentuk tabel dengan *field* dan *record*. Atribut menyatakan suatu parameter yang dibuat sebagai kriteria dalam pembentukan *tree*. Manfaat utama dari penggunaan pohon keputusan adalah kemampuannya untuk menyederhanakan proses pengambilan keputusan yang kompleks menjadi lebih sederhana sehingga pengambilan keputusan lebih menginterpretasikan solusi permasalahan. Pohon keputusan juga berguna untuk mengeksplorasi data, yaitu menemukan hubungan tersembunyi antara sejumlah calon variabel *input* dengan sebuah variabel target. Pohon keputusan merupakan himpunan aturan *IF...THEN*.

Setiap *path* dalam *tree* dihubungkan dengan sebuah aturan, dimana premis terdiri atas sekumpulan *node* yang ditemui dan kesimpulan aturan terdiri atas kelas yang terhubung dengan *leaf* dari *path*. *Decision tree* terdiri dari sekumpulan aturan untuk membagi sejumlah populasi yang heterogen menjadi lebih kecil, lebih homogen dengan memperhatikan variabel tujuannya. Variabel tujuan biasanya dikelompokkan dengan pasti dan model pohon keputusannya lebih mengarah kepada perhitungan probabilitas dari masing-masing *record* terhadap kategori atau untuk mengklasifikasikan *record* dengan mengelompokkannya dalam kelas yang sama.

Pohon keputusan merupakan salah satu metode klasifikasi yang menggunakan representasi struktur pohon *(tree)* dimana setiap simpul internal *(internal node)* merupakan sebuah atribut, setiap cabang merupakan nilai atribut, dan setiap simpul daun *(leaf node)* atau simpul terminal merupakan label *class*, serta simpul yang paling atas adalah simpul akar *(root node)*.

Elemen - elemen pada *decision tree* terdiri dari satu atau lebih variable atribut *(predictive attributes or features)* dan satu atribut kelas *(single class variable)*. *Classification tree* dapat digunakan untuk memprediksi nilai pada atribut kelas dengan mempertimbangkan nilai pada setiap variabel atribut yang diuji (Abellán & Castellano, 2017).



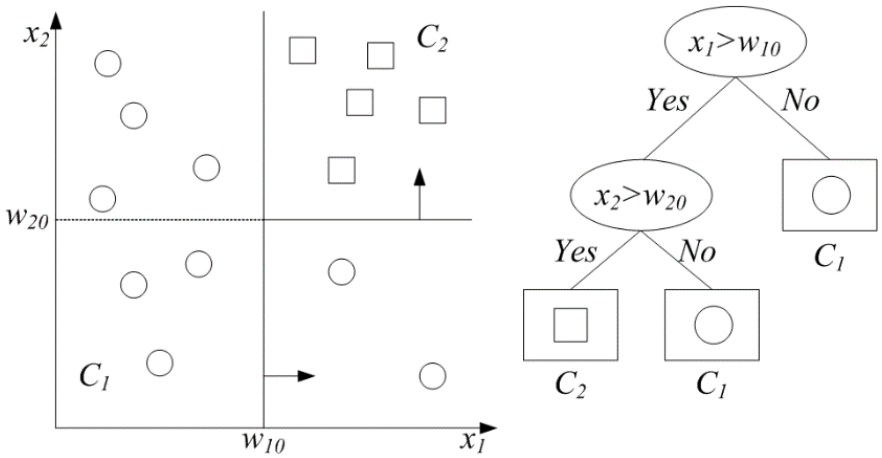
Gambar II.4 Skema *decision tree*

Berikut penjelasan mengenai 3 jenis simpul yang terdapat pada pohon keputusan diatas:

1. Simpul akar merupakan simpul yang paling atas. Simpul ini tidak mempunyai *input* dan mempunyai *output* lebih dari satu.
2. Simpul internal merupakan simpul percabangan dari simpul akar. Simpul ini hanya ada satu *input* dan mempunyai minimal dua *output*.
3. Simpul daun merupakan simpul terakhir. Simpul ini hanya terdapat satu *input* dan tidak ada keluaran *(output)*, simpul ini sering disebut juga simpul terminal.

Metode pohon keputusan mengubah fakta yang sangat besar menjadi pohon keputusan yang merepresentasikan aturan. Aturan dapat dengan mudah dipahami dengan bahasa alami. Dan juga dapat diekspresikan dalam bentuk Bahasa terstruktur yang disebut *Structured Query Language* untuk mencari *record* pada kategori tertentu. *Decision tree* berguna untuk mengeksplorasi data, menemukan hubungan tersembunyi antara sejumlah calon variabel *input* dengan sebuah variabel target. Karena *decision tree* memadukan antara eksplorasi data dan pemodelan,

decision tree juga sangat bagus sebagai langkah awal dalam pembentukan model bahkan ketika dijadikan sebagai model akhir dari beberapa teknik lain.



Gambar II.5 Pemetaan atribut dataset dan model klasifikasi *decision tree*

Sumber: (Alpaydın, 2010)

Gambar II.5 menjelaskan tentang pemetaan atribut sebuah *dataset* untuk dimodelkan pada pengklasifikasian *decision tree*. *Node* yang berbentuk oval adalah simpul akar *(root/internal node)* dan *node* berbentuk persegi adalah simpul daun/subset *(leaf node)*. Hasil perhitungan probabilitas dari *record* pada atribut x_1 lebih besar dari *record* pada atribut w_{10} sehingga dipilih sebagai simpul akar dan memiliki 2 simpul percabangan dengan kondisi nilai *Yes* dan *No*. Simpul cabang yang bernilai *No* diklasifikasikan kedalam subset C_1 sedangkan simpul cabang yang bernilai *Yes* diklasifikasikan kedalam subset C_2 namun belum termasuk ke dalam kelas yang sama sehingga dilakukan kembali perhitungan probabilitas secara rekursif untuk membentuk pohon subset dan menghasilkan kembali 2 simpul percabangan yang memiliki nilai *Yes* dan *No*, semua subset C_1 dan C_2 masuk ke dalam kelas yang sama, maka setelah dilakukan *split* pertumbuhan *tree* dihentikan.

II.9 Model *Decision Tree* C4.5

C4.5 adalah algoritma klasifikasi *supervised learning* untuk membentuk *decision tree* dari data yang dikembangkan oleh J. Ross Quinlan sebagai pengembangan dari algoritma ID3. Jika ID3 *(Iterative Dichotomiser 3)* menggunakan *entropy* untuk kriteria *split*, sedangkan C4.5 *decision tree* menggunakan kriteria *split* yang telah dimodifikasi yang dinamakan *Gain Ratio* (Mitchell, 1997) dalam proses pemilihan *split* atribut. *Split* atribut merupakan proses utama dalam pembentukan pohon keputusan *(decision tree)*. Algoritma C4.5 dapat bekerja pada variabel kontinyu dan *missing value* (Quinlan, 1986). Atribut yang memiliki *Gain Ratio* tertinggi yang dipilih (T. Wang et al., 2012).

C4.5 menggunakan dua pendekatan *heuristic* untuk dilakukan uji peringkat probabilitas yaitu:

1. *Information gain*, meminimalkan total *entropy* dari subset {Si} dimana terjadi bias saat diuji dengan data numerik.
2. *Gain ratio*, pembagian *information gain* dengan informasi *entropy* dari tiap atribut.

Agoritma C4.5 merupakan salah satu varian *decision tree* yang mirip dengan struktur *flowchart*, yang masing-masing internal node dinyatakan sebagai atribut pengujian. Setiap cabang mewakili *output* dari pengujian dan tiap *node (leaf node)* menentukan label *class*. Node paling atas dari sebuah pohon adalah *node root*. Algoritma C4.5 menggunakan *information gain* sebagai penentu simpul akar, internal dan daun. Tahapan algoritma C4.5 adalah sebagai berikut: (Mitchell, 1997)

1. Menghitung nilai Entropy pada masing – masing atribut:

$$Entropy\ (S) = \sum_{i=1}^{n} -pi \times log_2\ pi \quad (2.18)$$

dimana:

S = himpunan kasus

n = jumlah partisi *S*

pi = proporsi subset dari *S* pada partisi ke-*i*

2. Menghitung nilai Information Gain pada masing-masing atribut:

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \times Entropy(S_i) \quad (2.19)$$

dimana:

S = keseluruhan *dataset*

A = atribut *subset*

n = jumlah partisi atribut A

| *Si* | = ukuran subset dari *dataset* yang dimiliki atribut *A* pada partisi ke-*i*

| *S* | = ukuran jumlah kasus dalam *dataset*

3. Menghitung nilai Split Information *untuk masing-masing atribut:*

$$SplitInfo\ _A\ (D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2\ (\frac{|D_j|}{|D|}) \quad (2.20)$$

dimana:

D = keseluruhan *dataset*

A = atribut *subset*

v = jumlah partisi atribut *A*

| D_j | = ukuran subset dari *dataset* yang dimiliki atribut *A* partisi ke-*j*

| *D* | = ukuran jumlah kasus dalam *dataset*

4. Menghitung nilai *Gain Ratio* untuk masing-masing atribut:

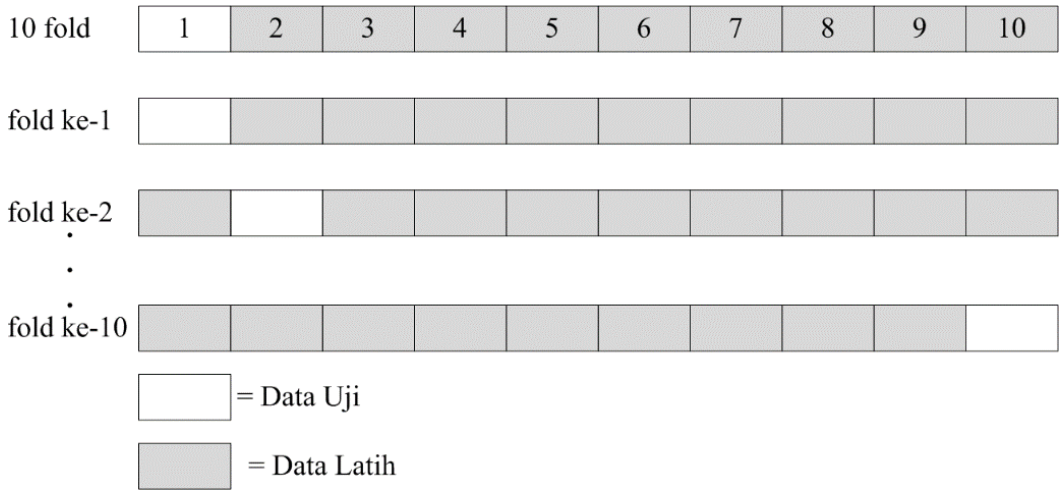
$$Gain\ Ratio\ (A) = -\frac{Gain\ (A)}{SplitInfo\ (A)} \quad (2.21)$$

5. Atribut yang memiliki *Gain Ratio* tertinggi dipilih menjadi akar *(splitting attribute)* dan atribut yang memiliki nilai *Gain Ratio* lebih rendah dari akar *(root)* dipilih menjadi cabang *(branches)*.
6. Menghitung lagi nilai *Gain Ratio* tiap-tiap atribut dengan tidak mengikutsertakan atribut yang terpilih menjadi akar *(root)* di tahap sebelumnya.
7. Atribut yang memiliki *Gain Ratio* tertinggi dipilih menjadi cabang *(branches)*.
8. Mengulangi langkah ke-4 dan ke-5 sampai dengan dihasilkan nilai *Gain* = 0 untuk semua atribut yang tersisa.

II.10 *K-Fold Cross Validation*

Sejak diperkenalkannya metode *leave-one-out Cross Validation* (LOOCV) oleh Stone pada tahun 1974, *cross validation* menjadi salah satu metode yang paling populer dalam mengevaluasi hasil kinerja algoritma. (Jung, 2018). Salah satu prosedur standar untuk mengevaluasi kinerja algoritma menggunakan *cross validation* adalah *k-fold cross validation*. *K-fold cross validation* diilustrasikan dalam membagi kumpulan data secara acak menjadi *k* lipatan dengan jumlah data sama pada setiap *k* lipatan. Setiap lipatan pada gilirannya akan digunakan untuk menguji model klasifikasi dari lipatan *k* lainnya (Wong & Yang, 2017). Tujuan dilakukannya *k-fold cross validation* adalah untuk memvalidasi algoritma klasifikasi yang digunakan lebih teruji dan kinerja yang dihasilkan bersifat valid.

Nilai *k* pada metode *k-fold cross validation* adalah sebuah bilangan bulat yang digunakan untuk membagi data. Maka data yang dihasilkan akan dibagi sebanyak *k* bagian dan metode klasifikasi akan melakukan proses pembelajaran dan pengujian sebanyak *k*. Selama iterasi prosedur klasifikasi sebanyak *k* kali, masing-masing bagian *k* akan ditetapkan sebagai data validasi (Jung, 2018). Ketika jumlah atribut dalam kumpulan data sangat besar, maka *k-fold cross validation* dapat menghasilkan akurasi yang lebih tinggi daripada metode validasi *hold-out validation*.



Gambar II.6 Contoh iterasi data dengan *cross validation*

Kinerja dari *k-fold cross validation* yaitu:

1. Total *instance* dibagi menjadi *N* bagian.
2. *Fold* ke-1 adalah ketika bagian ke-1 menjadi data uji (*testing* data) dan sisanya menjadi data latih (*training* data). Selanjutnya, hitung akurasi atau kesamaan atau kedekatan suatu hasil pengukuran dengan angka atau data

yang sebenarnya berdasarkan porsi data tersebut. Perhitungan akurasi tersebut menggunakan persamaan sebagai berikut:

$$Akurasi = \frac{\sum data\ uji\ benar\ klasifikasi}{\sum total\ data\ uji} \times 100\% \quad (2.22)$$

3. *Fold* ke-2 adalah ketika bagian ke-2 menjadi data uji (*testing* data) dan sisanya menjadi data latih (*training* data). Selanjutnya, hitung akurasi berdasarkan porsi data tersebut.
4. Demikian seterusnya hingga mencapai *fold* ke-*k*. Hitung rata-rata akurasi dari *k* buah akurasi di atas. Rata-rata akurasi ini menjadi akurasi final.

II.11 *Confusion Matrix*

Confusion matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep data mining. Pengukuran terhadap kinerja suatu sistem klasifikasi merupakan hal yang penting. Kinerja sistem klasifikasi menggambarkan seberapa baik sistem dalam mengklasifikasikan data. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya.

Confusion Matrix mempunyai dua dimensi yaitu dimensi yang berisi data aktual suatu objek dan dimensi yang berisi hasil prediksi Teknik klasifikasi. Tabel II.1 adalah contoh ilustrasi *confusion matrix*.

Tabel II.1 ***Confusion matrix***

		Prediction	
		True	**False**
True Value	**True**	True Positive (TP)	True Negative (TN)
	False	False Positive (FP)	False Negative (FN)

Sumber: (Q. Wang, 2014)

dimana:

TP = Jumlah data positif yang terklasifikasi dengan benar oleh sistem

TN = Jumlah data negatif yang terklasifikasi dengan benar oleh sistem

FN = Jumlah data negatif namun terklasifikasi salah oleh sistem

FP = Jumlah data positif namun terklasifikasi salah oleh sistem

Berdasarkan nilai TP,TN, FP *dan* FN *dapat diperoleh nilai metric yang biasa dipergunakan untuk menghitung kinerja dari model klasifikasi seperti* Accuracy, Precision, Recall/Sensitivity, Specificity, F1 Measure F1 Score dan AUC, *sebagai berikut:*

1. *Accuracy*

   Accuracy merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data.

$$Akurasi = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (2.23)$$

   tingkat akurasi dapat di diagnosa sebagai berikut (Gorunescu, 2011):

   Akurasi 0.90 – 1.00 = *Excellent classification*

   Akurasi 0.80 – 0.90 = *Good classification*

   Akurasi 0.70 – 0.80 = *Fair classification*

Akurasi 0.60 – 0.70 = *Poor classification*

Akurasi 0.50 – 0.60 = *Failure*

2. *Precision*

   Precision merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif.

$$Precision = \frac{(TP)}{(TP+FP)} \quad (2.24)$$

3. *Recall/Sensitivity*

   Recall/Sensitivity merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif.

$$Recall(Sensitivity) = \frac{(TP)}{(TP+FN)} \quad (2.25)$$

4. *Specificity*

   Specificity merupakan kebenaran memprediksi negatif dibandingkan dengan keseluruhan data negatif.

$$Specificity = \frac{(TN)}{(TN+FP)} \quad (2.26)$$

5. *F1 Measure*

   F1 Measure adalah perhitungan kombinasi antara *recall* dan *precision*. Rentang nilai *F1 Measure* adalah 0 sampai dengan 1, jika nilai mendekati 0 maka model prediksi tidak baik dan sebaliknya jika nilai mendekati 1 maka model prediksi baik. Untuk mendapatkan nilai dalam persentase maka nilai dikalikan dengan 100.

$$F1\ Measure = \frac{2 \times precision \times recall}{precision+recall} \quad (2.27)$$

6. *F1 Score*

F1 Score adalah untuk mengevaluasi seberapa baik *metric hybrid* yang dipergunakan untuk kelas tidak seimbang. Rentang nilai *F1 Score* adalah 0 sampai dengan 1, jika nilai mendekati 0 maka model prediksi tidak baik dan sebaliknya jika nilai mendekati 1 maka model prediksi baik. Untuk mendapatkan nilai dalam persentase maka nilai dikalikan dengan 100.

$$F1\ Score = \frac{(2TP)}{(2TP+FP+FN)} \tag{2.28}$$

7. *Receiver Operating Curve* (ROC)

   ROC merupakan *plot* dari *True Positive Rate* (TPR) dibandingkan dengan *False Positive Rate* dengan memvariasikan ambang batas. *Metric* ini sebagaimana yang diperlihakan pada table berikut ini:

   Tabel II.2 ROC

Metric	Formula	Equivalent
TPR	$\frac{(TP)}{(TP+FN)}$	Sensitivity
FPR	$\frac{(FP)}{(TN+FP)}$	1 - Specificity

8. *Area Under Curve* (AUC)

   AUC merupakan area dibawah kurva ROC yang menggambarkan probabilitas dengan variabel *sensitivity* dan *specificity*.

$$AUC = \frac{Sensitivity+Specificity}{2} \tag{2.29}$$

   nilai AUC:

   0.90 – 1.00 = *Excellent classification*

   0.80 – 0.90 = *Good classification*

0.70 – 0.80 = *Fair classification*

0.60 – 0.70 = *Poor classification*

< 0.60 = *Failure*

II.12 *T-Test*

T-Test atau uji t sampel berpasangan *(paired-samples t-test)* digunakan untuk membandingkan selisih dua rata-rata *(mean)* dari dua sampel yang berpasangan. Uji T berpasangan biasanya digunakan untuk menguji hipotesis penelitian apakah hasil setelah perbaikan lebih baik dari hasil sebelum perbaikan. Bentuk uji t sampel berpasangan akan menggunakan bentuk Uji t sampel berpasangan *(paired samples t-test)* dua sisi *(two tailed)* dengan hipotesis mana yang diterima.

Hipotesis (Nuryadi et al., 2017):

$H_0 = \mu_1 - \mu_2 = 0\ atau\ \mu_1 = \mu_2$

$H_a = \mu_1 - \mu_2 \neq 0\ atau\ \mu_1 \neq \mu_2$

H_a berarti bahwa selisih sebenarnya dari kedua rata-rata tidak sama dengan nol. Rumus *paired sample t-test*:

$$t_{hit} = \frac{\bar{D}}{SD/\sqrt{n}} \tag{2.30}$$

dimana:

t_{hit} = nilai t hitung

$\bar{D}$ = rata-rata selisih pengukuran 1 dan 2

SD = standar deviasi selisih pengukuran 1 dan 2

$SD = \sqrt{var}$

$var(s)^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

n = jumlah *sample*

Interpretasi:

1. Untuk menginterpretasikan uji t-test terlebih dahulu harus ditentukan :
    a. Nilai signifikansi α
    b. Df *(degree of freedom)*= N-k, khusus untuk *paired sample t-test* df = N-1
2. Bandingkan nilai t_{hit} dengan $t_{tab=\alpha;n-1}$
3. Apabila:

    $t_{hit} > t_{tab}$ → Berbeda secara signifikansi (H_0 ditolak)

    $t_{hit} < t_{tab}$ → Tidak berbeda secara signifikansi (H_0 diterima)

II.13 Penelitian Terdahulu

Penelitian yang pernah dilakukan sebelumnya dapat dilihat pada tabel II.3.

Tabel II.3 Penelitian terdahulu

No.	Peneliti dan Judul	Metode	Hasil Penelitian
1	Kavitha & Kannan (2016) *"An Efficient Framework for Heart Disease Classification using Feature Extraction and Feature Selection Technique in Data Mining"*	*Feature extraction* dengan teknik *principal component analysis* (PCA) dan *feature selection* menggunakan *wrapper approach* serta kriteria *split* dari algoritma C4.5 untuk digunakan sebagai metode untuk menghasilkan fitur-fitur yang relevan dari *dataset* UCI *heart dataset*	Metode yang diusulkan dapat menghasilkan fitur-fitur yang relevan sehingga meningkatkan efisiensi dan akurasi

Tabel II.3 Penelitian terdahulu (lanjutan)

No.	Peneliti dan Judul	Metode	Hasil Penelitian
2	Jyotsna Yadava & Khushwant Sehra (2018) *"Large Scale Dual Tree Complex Wavelet Transform Based Robust Features in PCA and SVD Subspace for Digital Image Watermarking"*	Skema *watermarking* baru menggunakan *feature extraction* dengan teknik *principal component analysis* (PCA) dan *singular value decomposition* (SVD) pada *dual tree complex wavelet transform* (DTCWT) di *low dimensional subspace* untuk gambar *grayscale*	Menghasilkan nilai *peak signal to noise ratio* (PSNR) dan korelasi yang tinggi dibandingkan dengan *state of art techniques* menunjukkan ketidakterlihatan dan ketahanan skema pada metode yang diusulkan
3	Said Bahassine, Abdellah Madani, Mohammed Al-Sarem dan Mohamed Kissi (2018) *"Feature Selection Using an Improved Chi-square for Arabic Text"*	*Feature selection* dengan teknik *chi-squared* pada klasifikasi *support vector machine* (SVM) untuk *text mining*	Metode yang diusulkan terbukti secara signifikan meningkatkan kinerja model klasifikasi teks arab
4	Tengku Mazlin Tengku Ab Hamid, Roselina Sallehuddin, Zuriahati Mohd Yunos dan Aida Ali (2021) *" Ensemble Based Filter Feature Selection with Harmonize"*	Pemilihan fitur *ensemble multi filter* yaitu *information gain, gain ratio, chi-square* dan *relief-f,* kemudian *support vector machine* digunakan untuk mengevaluasi kinerja klasifikasi fitur yang dipilih. Setelah fitur *ensemble* didapatkan, selanjutnya dilakukan optimasi dengan parameter klasifikasi secara sinkron dengan *particle swarm optimization* dan *support vector machine*. Pengujian menggunakan *dataset breast cancer* dan l*ymphography* dari UCI *machine learning repository*.	Hasil dari eksperimen menunjukkan bahwa metode yang diusulkan berhasil menunjukkan kinerja akurasi *classifier* dengan optimal fitur yang signifikan dibandingkan dengan metode lain yang ada seperti PSO-SVM dan *classical* SVM. Oleh karena itu, metode yang diusulkan dapat digunakan sebagai metode alternatif untuk menentukan solusi optimal dalam menangani data yang berdimensi tinggi.

Tabel II.3 Penelitian terdahulu (lanjutan)

No.	Peneliti dan Judul	Metode	Hasil Penelitian
5	Jixiong Zhang, Yanmei Xiong dan Shungeng Min (2019) *"A New Hybrid Filter/Wrapper Algorithm for Feature Selection in Classification"*	*Feature selection* dengan teknik *large margin hybrid algorithm for feature selection* (LMFS) yang merupakan *hybrid filter/wrapper algorithm*, pada klasifikasi menggunakan k-NN, PLS-DA dan LS-SVM untuk *dataset coffees, meats, two tablets, olive oils* dan *tobaccos* dari *quadram institute dataset*	Fitur yang dipilih oleh metode LMFS memiliki kinerja klasifikasi dan interpretasi model yang lebih baik. Selain itu, LMFS dapat secara efektif mengatasi dampak kompleksitas pengklasifikasi pada waktu komputasi dan *distance-based classifiers* ditemukan lebih cocok untuk memilih subset terakhir dalam LMFS
6	Letícia Silva, Bruno Bispo dan João Paulo Teixeira (2020) *"Features Selection Algorithms for Classification of Voice Signals"*	*Feature selection* dengan teknik *forward selection* dan *backward selection* pada implementasi klasifikasi *pearson's linear correlation, relieff, welch's t-test* dan algoritma berbasis *multilinear regression* untuk pemilihan *feature acoustic* identifikasi patologi suara	Dari metode-metode yang diusulkan didapat bahwa metode *relieff algorithm* dengan *feature selection* dapat memberikan performa terbaik dibanding metode lainnya

Tabel II.3 Penelitian terdahulu (lanjutan)

No.	Peneliti dan Judul	Metode	Hasil Penelitian
7	Parnika Bhat, Kamlesh Dutta (2021) *"A Multi-tiered Feature Selection Model for Android Malware Detection Based on Feature Discrimination and Information Gain"*	*feature discrimination* dan *information gain* pada pemilihan fitur *multi-tier model*, yang dapat menemukan fitur yang relevan dan signifikan untuk meningkatkan akurasi pendeteksian *malware*. Kemudian kumpulan fitur yang dipilih diaplikasikan pada lima teknik klasifikasi *machine learning*. Pengujian dilakukan pada *dataset Virustotal, VirusShare* dan *Drebin* yang diperoleh dari pihak ketiga.	Pengujian dan analisis yang ketat menunjukkan bahwa klasifikasi *Random Forest* mencapai tingkat akurasi tertinggi sebesar 96,28%.
8	Syed Javeed Pasha, E. Syed Mohamed (2022) *"Advanced Hybrid Ensemble Gain Ratio Feature Selection Model Using Machine Learning for Enhanced Disease Risk Prediction"*	*Advanced hybrid ensemble gain ratio feature selection* (AHEG-FS) pada *machine learning* untuk meningkatkan prediksi risiko penyakit jantung, teknik pemilihan fitur yang digunakan yaitu: *ensemble feature selection, gain ratio feature selection* dan *backward feature elimination*, kemudian menggunakan *area under the curve* (AUC) sebagai evaluasi untuk prediksi risiko penyakit yang efektif. Pengujian dilakukan pada 4 *dataset Irvine M*L *repository* dari *University of California* yaitu: *heart disease datasets Cleveland, Hungarian, Statlog* dan *Switzerland*.	Dari penelitian didapati bahwa *backward feature elimination* menghasilkan perolehan subset terbaik dari fitur yang sangat berkontribusi dan menghasilkan presisi tertinggi.

Tabel II.3 Penelitian terdahulu (lanjutan)

No.	Peneliti dan Judul	Metode	Hasil Penelitian
9	Deepak Kshirsagar, Sandeep Kumar (2021) *"An Efficient Feature Reduction Method for the Detection of DoS Attack"*	reduksi fitur berdasarkan kombinasi algoritma reduksi fitur berbasis filter, yaitu *information gain ratio* (IGR), *correlation* (CR), dan *relieff* (ReF) pada *dataset* CICIDS 2017 DoS dan KDD *Cup* 99. Tujuan penelitian ini adalah untuk peningkatan kinerja dengan mengurangi jumlah fitur untuk mendeteksi serangan DoS.	Dari penelitian didapati bahwa pengurangan fitur yang diusulkan dapat mengurangi jumlah fitur CICIDS 2017 dan KDD *Cup* 99 dari 77 menjadi 24, dan 41 menjadi 12, menghasilkan peningkatan akurasi 99,9593% menggunakan pengklasifikasi *projective adaptive resonance theory* (PART) dengan waktu proses 133,66 s pada *dataset* CICIDS 2017. metode ini juga menghasilkan akurasi yang signifikan pada *dataset* KDD *Cup* 99.
10	Anna Karen Garate-Escamila, Amir Hajjam El Hassani, Emmanuel Andres (2020) *"Classification Models for Heart Disease Prediction Using Feature Selection and PCA"*	*dimensionality reduction chi-square*, *principal component analysis* (PCA) dan *chi-square* dan *principal component analysis* (CHI-PCA) dengan tujuan untuk menemukan ciri penyakit jantung dengan menerapkan teknik seleksi ciri pada *dataset heart disease* dari UCI *machine learning repository*, kemudian divalidasi menggunakan enam jenis pengklasifikasian pada *machine learning*.	Metode *chi-square* dan *principal component analysis* (CHI-PCA) dengan *random forests* (RF) memiliki akurasi tertinggi, dengan 98,7% untuk *dataset Cleveland*, 99,0% untuk *dataset Hungaria*, dan 99,4% untuk *dataset Cleveland-Hungaria* (CH).

Tabel II.3 Penelitian terdahulu (lanjutan)

No.	Peneliti dan Judul	Metode	Hasil Penelitian
11	Pushparaj Nimbalkar, Deepak Kshirsagar (2021) *"Feature Selection for Intrusion Detection System in Internet-of-Things (IoT)"*	pemilihan fitur untuk sistem deteksi intrusi (IDS) menggunakan *information gain* (IG) dan *gain ratio* (GR) dengan peringkat 50% fitur teratas untuk mendeteksi serangan DoS dan DDoS, kemudian dievaluasi dan divalidasi dengan *JRipclassifier*. Pengujian menggunakan *dataset* IoT-BoT dan KDD *Cup* 1999.	Sistem mencapai akurasi dan tingkat deteksi yang lebih tinggi masing-masing 99,9993%, dan 99,5798%, dengan *JRip* menggunakan 16 fitur pada *dataset* BoT-IoT. Validasi sistem *dataset* KDD *Cup* 1999 juga meningkatkan akurasi dan tingkat deteksi 99,9920% dan 99,9943% untuk mendeteksi serangan DoS menggunakan 19 fitur dengan *JRip*.