BAB II
TINJAUAN PUSTAKA

II.1 Klasifikasi

Klasifikasi adalah suatu metode analisis data yang dapat membantu orang menentukan label kelas dari sampel yang ingin diklasifikasi. Klasifikasi merupakan suatu metode *supervised* learning dimana metode ini mencoba untuk menemukan hubungan antara antribut masukan dan atribut target dengan tujuan untuk meningkatkan kehandalan hasil yang diperoleh dari suatu data (Alnuaimi & Albaldawi, 2024). Dalam proses ini, model yang telah dilatih mampu mengklasifikasikan data baru secara otomatis berdasarkan pola yang telah dipelajari.

Klasifikasi juga dapat dikatakan sebagai suatu metode pengelompokkan data yang telah mempunyai kelas sebelumnya. Berdasarkan data-data histori akan terbentuk sebuah *rule* yang dapat figunakan untuk menentukan kelas dari data berikutnya (Ramadandi, S, 2020). Biasanya, proses klasifikasi melibatkan tahap pelatihan (training) dan pengujian (testing) untuk menilai sejauh mana model dapat menggeneralisasi ke data baru.

Berbagai algoritma klasifikasi telah berkembang, seperti decision tree, Naive Bayes, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), dan ensemble learning seperti Random Forest. Setiap algoritma memiliki kelebihan dan kekurangan tergantung jenis dan karakteristik data. Misalnya, Random Forest sangat efektif dalam menangani data dengan fitur tinggi dan kompleksitas interaksi antar fitur (Zhao & Ye, 2024). Kualitas data sangat memengaruhi kinerja klasifikasi: data yang mengandung noise, outlier, atau distribusi tidak seimbang (imbalance) dapat menurunkan akurasi model. Oleh karena itu, strategi praproses seperti normalisasi, penghapusan duplikat, serta teknik oversampling atau undersampling sering diterapkan sebelum pelatihan model (Bommert et al., 2022). Evaluasi hasil klasifikasi biasanya dilakukan dengan metrik seperti akurasi, precision, recall, dan F1-score untuk mendapatkan gambaran menyeluruh tentang performa model.

Salah satu tantangan utama dalam klasifikasi adalah overfitting, yaitu kondisi ketika model terlalu menyesuaikan diri dengan data pelatihan sehingga kehilangan kemampuan untuk menggeneralisasi pada data baru. Overfitting sering terjadi ketika model terlalu kompleks atau data pelatihan tidak mencukupi. Untuk mengatasi hal ini, digunakan teknik validasi silang (cross-validation) dan regularisasi, seperti L1 (Lasso) dan L2 (Ridge), yang membantu mengurangi kompleksitas model agar tetap seimbang.

Di sisi lain, underfitting terjadi ketika model tidak cukup kompleks untuk menangkap pola dari data. Hal ini menyebabkan performa rendah baik pada data pelatihan maupun data pengujian. Untuk menghindari underfitting, digunakan model dengan kapasitas lebih tinggi atau menambahkan fitur-fitur relevan dalam proses pelatihan. Penyesuaian parameter model (hyperparameter tuning) juga penting agar model mencapai performa optimal.

Pemilihan fitur (feature selection) merupakan langkah penting dalam klasifikasi, karena fitur yang tidak relevan atau redundan dapat memperburuk performa model. Metode seperti Relief-F, Mutual Information, dan Recursive Feature Elimination (RFE) digunakan untuk memilih fitur terbaik. Pemilihan fitur yang tepat tidak hanya meningkatkan akurasi tetapi juga mengurangi waktu komputasi dan meningkatkan interpretabilitas model.

Selain pemilihan fitur, ekstraksi fitur (feature extraction) juga memainkan peran penting, terutama untuk data seperti citra atau sinyal. Teknik seperti Principal Component Analysis (PCA), t-SNE, atau autoencoder digunakan untuk mengurangi dimensi data tanpa kehilangan informasi penting. Dengan cara ini, model klasifikasi dapat lebih efisien dan efektif dalam memproses data. Dalam konteks data tidak seimbang (imbalanced data), di mana satu kelas mendominasi kelas lain, diperlukan strategi khusus. Beberapa pendekatan yang umum digunakan adalah oversampling pada kelas minoritas (misalnya SMOTE) atau undersampling pada kelas mayoritas. Algoritma khusus seperti Balanced Random Forest dan Cost-Sensitive Learning juga dapat diterapkan untuk menangani masalah ini.

Penerapan klasifikasi sangat luas dan mencakup berbagai bidang, mulai dari kesehatan, keuangan, pemasaran, hingga keamanan siber. Dalam bidang medis,

klasifikasi digunakan untuk mendiagnosis penyakit berdasarkan citra medis atau data sensor, sedangkan di bidang keuangan, klasifikasi membantu dalam mendeteksi penipuan transaksi. Di industri e-commerce, klasifikasi digunakan untuk memprediksi ketertarikan pelanggan terhadap produk tertentu.

Salah satu perkembangan terbaru dalam klasifikasi adalah automated machine learning (AutoML) yang memungkinkan proses pemilihan model, tuning parameter, dan praproses dilakukan secara otomatis. Dengan AutoML, pengguna tanpa latar belakang teknis yang kuat tetap dapat membangun model klasifikasi yang kompetitif. Platform seperti Google AutoML dan Auto-Sklearn merupakan contoh implementasi AutoML yang sudah banyak digunakan.

Explainable AI (XAI) juga menjadi perhatian penting dalam klasifikasi modern. XAI bertujuan untuk memberikan penjelasan atas prediksi yang dihasilkan oleh model klasifikasi, terutama pada model yang bersifat "black box" seperti deep neural networks. Teknik seperti LIME, SHAP, dan Grad-CAM digunakan untuk menjelaskan kontribusi fitur terhadap prediksi yang dihasilkan model, yang sangat penting untuk aplikasi di bidang sensitif seperti medis dan hukum.

Evaluasi performa model klasifikasi tidak hanya berdasarkan akurasi semata. Dalam banyak kasus, terutama pada data tidak seimbang, metrik seperti AUC-ROC, confusion matrix, precision-recall curve, dan Matthews Correlation Coefficient (MCC) memberikan gambaran yang lebih lengkap. Oleh karena itu, dalam praktiknya, kombinasi berbagai metrik evaluasi digunakan untuk menilai model secara menyeluruh.

Dengan terus berkembangnya teknologi, klasifikasi berbasis pembelajaran federatif (federated learning) menjadi tren baru. Dalam pendekatan ini, model dilatih secara terdistribusi di banyak perangkat tanpa perlu mengirim data mentah ke server pusat, menjaga privasi pengguna. Ini sangat relevan di era IoT dan big data, di mana keamanan dan privasi menjadi pertimbangan utama dalam pemrosesan data.

Dalam era deep learning dan big data, klasifikasi semakin banyak digunakan menggunakan teknik seperti jaringan saraf tiruan dan transfer learning, yang memungkinkan model menangkap pola kompleks dari data terstruktur maupun

tidak (teks, gambar, sinyal, dsb.). Misalnya, arsitektur deep learning dapat digunakan untuk klasifikasi citra atau teks secara otomatis, sering kali menghasilkan akurasi yang lebih tinggi dibanding metode tradisional (Ahmad et al., 2024).

II.2 Naïve Bayes Classifier

Naive Bayes Classifier adalah suatu teknik dari statistik yang popular digunakan pemfilteran email. Metode ini muncul di pertengahan tahun 1990 dan merupakan salah satu upaya pertama untuk mengatasi masalah penyaringan spam. Metode ini termasuk metode yang sederhana namun memiliki performa yang baik dalam klasifikasi sentiment dan metode ini merupakan metode kedua terbanyak yang digunakan dalam analisis sentiment (Ligthart et al., 2021).

Model probabilistik *Naive Bayes* untuk model *classifier* adalah model bersyarat *P(C|f1,f2, ,fn)* atas variabel kelas dependen dengan jumlah hasil yang kecil, tergantung pada beberapa variable fitur *f1* hingga *fn*. Untuk sejumlah besar fitur reformulasi model dilakukan untuk membuatnya lebih mudah diatur.

Secara umum, *Naive Bayes Classifier* memiliki kinerja yang baik jika dibandingkan dengan pengklasifikasian lain karena kesederhanaannya, kompleksitas waktu yang lebih sedikit, kebutuhan memori yang kecil, dan akurasi prediksi yang baik.

Naive Bayes Classifier dapat dinyatakan dalam persamaan (2.1)

$$P(Ci|X) = \frac{P(X|Ci)P(Ci)}{\mathrm{P(X)}} \quad \text{.....(2.1)}$$

Keterangan:

Ci = Hipotesis data X yang merupakan kelas yang spesifik.

X = Data dengan kelas yang belum diketahui.

P(Ci|X) = Probabilitas hipotesis Ci berdasarkan kondisi X.

P(X|Ci) = Probabilitas X berdasarkan kondisi pada hipotesis Ci.

P(Ci) = Probabilitas hipotesis Ci.

P(X) = Probabilitas dari data X.

Dimana, probabilitas posterior *P(C|X)* (probabilitas dari nilai atribut *X* milik kelas *C)*, dihitung menggunakan *Class Prior Probability P(C)* (probabilitas kelas).

Predictior Prior Probability P(X) (probabilitas nilai atribut) dan *Likelihood P(X|C)* (probabilitas nilai atribut X yang diberikan kelas C). Dalam *Naive Bayes Classifier*, probabilitas masing-masing atribut berkenaan dengan kelas independen dari semua nilai atribut lainnya yang sangat kuat dan menyederhanakanperhitungan probabilitas, yang disebut sebagai *Conditional Independence.*

Meskipun asumsi "independen bersyarat" dalam Naive Bayes terdengar cukup ketat yaitu setiap fitur dianggap tidak saling bergantung jika diketahui kelasnya kenyataannya, dalam banyak kasus dunia nyata, asumsi ini tetap bekerja cukup baik. Ini salah satu alasan mengapa algoritma ini masih banyak digunakan, bahkan ketika data sebenarnya menunjukkan ketergantungan antar fitur. Salah satu contoh penerapan paling terkenal dari Naive Bayes adalah dalam sistem penyaringan spam email. Misalnya, jika kata "gratis" sering muncul dalam email spam, maka keberadaan kata tersebut akan memberikan bobot besar terhadap kemungkinan email itu termasuk spam. Dengan mengumpulkan statistik seperti ini dari banyak email, sistem dapat belajar membedakan mana yang spam dan mana yang bukan.

Naive Bayes juga sangat efektif untuk masalah klasifikasi teks lainnya, seperti analisis sentimen. Misalnya, algoritma ini dapat digunakan untuk memprediksi apakah sebuah ulasan film bersifat positif atau negatif, hanya berdasarkan kata-kata yang digunakan dalam ulasan tersebut. Kata-kata seperti "luar biasa" atau "membosankan" akan memiliki kontribusi besar terhadap klasifikasi akhir. Dalam pengolahan data besar, kecepatan dan efisiensi adalah kunci. Naive Bayes menonjol di sini karena algoritmanya ringan dan bisa dijalankan cepat bahkan untuk dataset yang besar. Tidak seperti metode pembelajaran lain yang memerlukan banyak iterasi atau tuning, Naive Bayes bekerja baik "langsung dari kotak" (out-of-the-box).

Meskipun sederhana, Naive Bayes juga fleksibel. Ada beberapa varian seperti Gaussian Naive Bayes (untuk data numerik yang diasumsikan mengikuti distribusi normal), Multinomial Naive Bayes (sering digunakan dalam klasifikasi dokumen), dan Bernoulli Naive Bayes (digunakan ketika fitur adalah biner, seperti kehadiran atau ketidakhadiran sebuah kata). Namun demikian, penting untuk

diketahui bahwa Naive Bayes memiliki keterbatasan. Karena algoritma ini mengasumsikan bahwa semua fitur independen terhadap satu sama lain, performanya bisa menurun jika fitur-fitur tersebut saling berkorelasi kuat. Misalnya, dalam data medis, tekanan darah dan kadar gula mungkin berkaitan, sehingga asumsi independensi tidak valid.

Kelebihan lain dari Naive Bayes adalah transparansi modelnya. Kita dapat dengan mudah melihat kontribusi masing-masing fitur terhadap prediksi akhir, sehingga model ini juga cocok dalam konteks yang menuntut interpretasi atau penjelasan, seperti dalam bidang hukum, keuangan, atau kesehatan. Naive Bayes juga menjadi algoritma favorit dalam pembelajaran mesin awal karena kemudahannya dipahami. Mahasiswa atau pemula dalam data science dapat dengan cepat mengimplementasikan dan menguji model ini hanya dengan beberapa baris kode. Banyak platform seperti Weka, Scikit-learn, dan RapidMiner sudah menyediakan implementasi siap pakai.

Menariknya, dalam beberapa kompetisi data science, para peserta menemukan bahwa model Naive Bayes sering menjadi baseline yang tangguh. Artinya, meskipun peserta menggunakan model yang lebih kompleks seperti Random Forest atau XGBoost, model Naive Bayes tetap memberikan akurasi yang mengejutkan untuk masalah klasifikasi sederhana. Kesimpulannya, Naive Bayes Classifier adalah alat yang sederhana namun sangat kuat, terutama dalam klasifikasi berbasis teks. Meskipun bukan solusi sempurna untuk semua masalah klasifikasi, kecepatan, efisiensi, dan akurasi yang cukup baik menjadikannya algoritma yang tak lekang oleh waktu, bahkan di era big data dan deep learning seperti sekarang.

II.3 *Relief-F*

Relief-F tetap menjadi algoritma populer untuk seleksi fitur berbasis filter karena mampu menilai relevansi fitur melalui pendekatan instance-based. Baru-baru ini banyak penelitian mengadaptasinya ke domain data modern seperti genomik dan citra hiperspektral. Misalnya, dalam pemetaan litologi, Relief-F dikombinasikan dengan Random Forest untuk mendapatkan subset fitur yang paling informatif.

Pengembangan model Relief-F untuk klasifikasi citra hiperspektral. Mereka

mengombinasikan Relief-F sebagai tahap filter awal dengan Random Forest untuk akurasi klasifikasi yang ditingkatkan. Hasilnya menunjukkan bahwa pendekatan itu efektif dalam mengurangi dimensi dan meningkatkan ketepatan identifikasi litologi (Xi et al., 2023).

Dalam domain biologi molekuler, sebuah studi menggunakan Relief-F untuk menyeleksi miRNA serum sebagai biomarker kanker paru. Dataset GEO dianalisis, lalu Relief-F menyaring kandidat miRNA sebelum membangun model klasifikasi berbasis SVM dan Random Forest. Model terbaik menunjukkan AUC sebesar 0,968, menandakan kinerja diagnostik yang sangat tinggi.

Relief-F merupakan pengembangan dari algoritma Relief yang mampu menangani data dengan banyak kelas dan data yang mengandung missing value. Algoritma ini bekerja dengan menilai pentingnya sebuah fitur berdasarkan seberapa baik fitur tersebut membedakan antara instance tetangga yang termasuk dalam kelas yang sama (nearest hit) dan kelas yang berbeda (nearest miss). Pendekatan berbasis instance ini menjadikan Relief-F lebih adaptif dibanding metode filter konvensional lain seperti chi-square atau Information Gain.

Kelebihan utama Relief-F terletak pada kemampuannya mempertahankan informasi interaksi antar fitur tanpa harus membangun model prediktif terlebih dahulu. Hal ini membuatnya sangat sesuai digunakan pada tahap pra-pemodelan untuk menyaring fitur-fitur tidak relevan, terutama dalam kondisi high-dimensional low-sample size seperti pada data genomik atau ekspresi gen. Relief-F juga berhasil diimplementasikan dalam klasifikasi citra medis. Dalam studi analisis citra histopatologi, algoritma ini digunakan untuk memilih fitur tekstur dan warna yang paling representatif dari ribuan kandidat fitur. Dengan kombinasi CNN dan Relief-F, studi ini berhasil menurunkan dimensi fitur hingga 80% tanpa kehilangan performa klasifikasi kanker.

Di bidang keamanan siber, Relief-F telah dimanfaatkan untuk analisis dan deteksi intrusi jaringan. Dengan menyeleksi fitur-fitur trafik jaringan yang paling relevan, sistem deteksi berbasis Relief-F dan deep learning dapat secara efisien membedakan aktivitas normal dan serangan seperti DoS, probing, dan R2L, serta menghasilkan deteksi dengan F1-score lebih dari 95%.Dalam pengembangan

sistem rekomendasi, Relief-F juga digunakan untuk menyaring atribut pengguna dan item yang paling berpengaruh terhadap preferensi. Dengan menggabungkan teknik ini ke dalam arsitektur hybrid filtering, sistem rekomendasi mampu meningkatkan personalisasi tanpa memerlukan data pengguna yang terlalu banyak.

Pengembangan algoritma turunan dari Relief-F seperti SURF, MultiSURF, dan Relief-F* juga semakin memperluas fleksibilitas penggunaannya. Misalnya, MultiSURF mengadopsi threshold adaptif dalam memilih tetangga untuk meningkatkan sensitivitas terhadap interaksi fitur non-linear. Hal ini berguna dalam domain neuroinformatika dan prediksi gangguan psikologis dari data kognitif.Kombinasi Relief-F dengan algoritma optimasi metaheuristik juga menjadi pendekatan populer saat ini. Sebagai contoh, kombinasi Relief-F dan Particle Swarm Optimization (PSO) digunakan untuk memilih subset fitur optimal dalam diagnosis diabetes. Pendekatan ini mampu meningkatkan sensitivitas dan spesifisitas model deteksi hingga lebih dari 90% dibanding model baseline tanpa seleksi fitur.

Adapun langkah-langkah dalam penerapan Relief-F adalah sebagai berikut:

1. Inisiasi

   Pada tahap awal algoritma Relief-F, kita terlebih dahulu menentukan beberapa parameter penting. Salah satunya adalah jumlah iterasi (m), yaitu seberapa banyak sampel data yang akan dipilih secara acak untuk dianalisis selama proses seleksi fitur. Jumlah iterasi ini bisa disesuaikan, tetapi biasanya nilainya sama dengan jumlah total data atau lebih kecil, tergantung pada kebutuhan dan efisiensi komputasi.

   Selain itu, kita juga harus menentukan nilai k, yaitu jumlah tetangga terdekat (nearest neighbors) yang akan digunakan dalam perhitungan. Tetangga ini terdiri dari data yang paling mirip dengan sampel acak yang sedang dianalisis baik dari kelas yang sama maupun berbeda. Nilai k ini sangat penting karena memengaruhi seberapa baik algoritma Relief-F dapat mengenali fitur yang benar-benar relevan.

   Setelah menentukan nilai m dan k, langkah selanjutnya adalah menginisialisasi bobot semua fitur, di mana setiap fitur diberikan nilai awal

nol. Nilai ini nantinya akan diperbarui sepanjang iterasi, berdasarkan seberapa besar kontribusi fitur tersebut dalam membedakan antara sampel dari kelas yang sama dan berbeda. Dengan kata lain, fitur yang dianggap penting akan mendapatkan bobot lebih tinggi setelah proses ini berjalan.

2. Loop untuk setiap iterasi

   Setelah tahap inisialisasi selesai, algoritma Relief-F akan masuk ke proses iteratif, di mana setiap iterasi akan menganalisis satu sampel data. Pada setiap iterasi ke-i, algoritma akan secara acak memilih satu data dari seluruh dataset. Data ini disebut sebagai sampel target dan diberi notasi R_i. Sampel target ini akan menjadi acuan untuk menilai seberapa penting setiap fitur dalam membedakan data dari kelas yang sama atau berbeda.

   Setelah memilih R_i, langkah selanjutnya adalah mencari tetangga terdekat (nearest neighbors) dari sampel tersebut. Tetangga terdekat ini dibagi menjadi dua jenis:

   Nearest hits: yaitu k data lain yang paling mirip dengan R_i dan berasal dari kelas yang sama. Mereka disebut HitNeighbors, dan digunakan untuk mengukur seberapa konsisten nilai fitur dalam satu kelas.

   Nearest misses: yaitu k data lain yang paling mirip dengan R_i tetapi berasal dari kelas yang berbeda. Mereka disebut *MissNeighbors*, dan berfungsi untuk mengukur seberapa baik fitur dapat membedakan kelas.

   Dengan membandingkan nilai-nilai fitur dari nearest hits dan *nearest misses* terhadap sampel target, algoritma akan mulai memperbarui bobot fitur berdasarkan kontribusinya dalam mempertahankan atau membedakan kelas. Proses ini diulang untuk setiap iterasi, sampai seluruh proses seleksi fitur selesai.

3. Memperbaharui bobot fitur

   Setelah menemukan tetangga terdekat (baik yang berasal dari kelas yang sama maupun berbeda), algoritma Relief-F akan mulai memperbarui bobot setiap fitur. Tujuannya adalah untuk menilai seberapa besar kontribusi masing-masing fitur dalam membedakan kelas data. Untuk setiap fitur A, bobotnya akan diperbarui berdasarkan dua hal: perbedaan nilai fitur antara

sampel target dan tetangga dari kelas yang sama (nearest hits), serta perbedaan nilai fitur antara sampel target dan tetangga dari kelas yang berbeda (nearest misses).

Jika nilai fitur antara sampel target dan tetangga dari kelas yang sama banyak berbeda, maka fitur tersebut dianggap kurang baik karena tidak konsisten di dalam satu kelas sehingga bobotnya akan dikurangi. Sebaliknya, jika nilai fitur berbeda secara konsisten antara kelas yang berbeda, maka fitur dianggap bagus untuk membedakan kelas dan bobotnya akan ditambahkan. Besarnya perbedaan diukur menggunakan fungsi yang disebut diff, yaitu selisih nilai fitur yang dinormalisasi. Untuk data numerik, selisihnya dihitung dengan mengurangi dua nilai dan membaginya dengan rentang maksimum-minimum dari fitur tersebut. Sementara untuk data kategorik, perbedaan diberi nilai 1 jika berbeda, dan 0 jika sama. Bobot akan terus diperbarui sepanjang iterasi dengan mempertimbangkan proporsi kelas yang berbeda melalui probabilitas $P(c)$.

4. Normalisasi dan seleksi fitur

   Setelah semua iterasi selesai, setiap fitur akan memiliki nilai bobot akhir yang mencerminkan tingkat kepentingannya dalam membedakan data berdasarkan kelas. Fitur-fitur dengan nilai bobot tertinggi dianggap sebagai fitur yang paling relevan atau informatif. Pada tahap akhir ini, pengguna dapat memilih beberapa fitur teratas berdasarkan nilai bobot tersebut untuk digunakan dalam proses pelatihan model atau analisis lebih lanjut. Dengan demikian, proses ini membantu menyaring fitur-fitur yang tidak penting, mengurangi kompleksitas data, dan meningkatkan kinerja algoritma pembelajaran mesin

II.4 *Gain Ratio*

Gain ratio merupakan modifikasi dari *Information Gain* yang mengurangi biasnya. *Gain ratio* memperbaiki information gain dengan mengambil informasi intrinsik dari setiap atribut. *Gain ratio* adalah salah satu teknik dalam melakukan seleksi fitur yang menggunakan nilai *information gain* untuk dimodifikasi dalam mengurangi efek bias pada hasil nilai *information gain*. Metode seleksi fitur *gain*

ratio berpengaruh dalam meningkatkan akurasi dari suatu algortima salah satunya adalah algoritma KNN (Sari et al., 2023).

Adapun langkah-langkah dalam *gain* ratio adalah:

1. *Entropy* kumpulan data adalah jumlah nilai yang dibutuhkan untuk menyatakan atribut. Pengurangan yang diharapkan dalam *entropy* disebabkan oleh partisi sesuai dengan atribut. Hitung nilai *entropy* kumpulan data pada masing – masing atribut, dengan persamaan:

$$\text{Entropy}\ (S) = \sum_{i=1}^{n} - pi * log_2 pi \quad \ldots(2.2)$$

Di mana:

S = Himpunan Kasus

N = Jumlah Partisi S

p_i = Proporsi dari S_i terhadap S

2. *Information gain* adalah salah satu salah satu atribut seleksi yang digunakan untuk memilih pengujian atribut setiap *node* pada *tree*. Atribut dengan *information gain* tertinggi dipilih sebagai pengujian atribut dari suatu *node*. Hitung nilai *information gain* pada masing-masing atribut dengan persamaan:

$$Information\ Gain\ (S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \times Entropy(S_i) \quad \ldots(2.3)$$

Di mana:

S : Keseluruhan Dataset

A : Atribut Subset

N : Jumlah Partisi Atribut A

| Si | : Ukuran Subset dari Dataset yang dimiliki atribut pada A partisi ke-i

| S | : Ukuran Jumlah Kasus dalam Dataset

3. Jika nilai *Split Information* mendekati 0, rasio menjadi tidak stabil. Dimana nilai *information gain* dari pengujian yang dipilih harus besar, setidaknya sama besar dengan rata-rata gain dari semua pengujian yang telah diperiksa. Hitung nilai *Split Information* untuk masing-masing atribut dengan persamaan berikut:

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2(\frac{D_j}{D}) \quad \ldots(2.4)$$

Di mana:

D : Keseluruhan Dataset

A : Atribut Subset

V : Jumlah Partisi Atribut A

| Dj | : Ukuran Subset dari Dataset yang dimiliki atribut pada A partisi ke-j

| D | : Ukuran Jumlah Kasus dalam Dataset

4. Parameter menormalisasi nilai *information gain* dan tidak membiarkan bias pada atribut tersebut. Normalisasi dilakukan dengan membagi information gain atribut oleh *SplitInfo*, hasil nilai tersebut dikenal sebagai *gain ratio* dari suatu atribut. Berikut perhitungan *gain ratio* dari setiap atribut dengan menggunakan persamaan di bawah ini:

$$Gain\ Ratio\ (A) = \frac{Gain\ (A)}{SplitInfo(A)} \quad \text{....(2.5)}$$

Semakin banyak *information gain* didapat, maka akan semakin kuat atributnya. Maka, partisi akan terjadi pada atribut yang memiliki perolehan informasi tertinggi. *Gain Ratio* merupakan pengembangan dari *information gain*, dimana *gain ratio* menghilangkan nilai bias dari setiap atribut.

Gain Ratio dikembangkan sebagai solusi terhadap kelemahan utama dari Information Gain. Masalah utama yang dihadapi Information Gain adalah kecenderungannya untuk lebih memilih atribut dengan banyak nilai unik, bahkan jika atribut tersebut tidak benar-benar relevan. Sebagai contoh, atribut seperti "nomor ID pelanggan" bisa memiliki nilai Information Gain tinggi karena nilai yang unik, padahal sebenarnya atribut ini tidak memiliki informasi penting untuk klasifikasi. Di sinilah Gain Ratio memainkan peran penting. Dengan menambahkan proses normalisasi melalui nilai Split Information, Gain Ratio mencegah pemilihan atribut yang bersifat “unik tapi tidak berguna”. Split Information sendiri mengukur seberapa banyak partisi yang dihasilkan oleh atribut, dan kemudian digunakan untuk membagi nilai Information Gain agar nilainya menjadi proporsional terhadap informasi yang benar-benar berguna.

Dalam praktiknya, Gain Ratio sering digunakan dalam algoritma seperti C4.5 dan algoritma pohon keputusan lainnya. Alih-alih hanya mengejar atribut dengan IG (Information Gain) tertinggi, algoritma ini akan memilih atribut yang memiliki Gain Ratio tertinggi. Hal ini membuat model lebih stabil dan tidak mudah terjebak pada atribut palsu yang tampak menarik karena keunikannya saja. Misalnya, dalam kasus pengklasifikasian apakah seorang pasien berisiko terkena

penyakit tertentu, atribut seperti “jumlah kunjungan ke dokter” mungkin tampak relevan jika dilihat dari Information Gain. Namun, dengan perhitungan Gain Ratio, bisa saja atribut seperti “riwayat keluarga penyakit” justru lebih informatif dan konsisten di seluruh partisi data.

Salah satu kekuatan Gain Ratio adalah kemampuannya dalam membantu pemilihan fitur yang benar-benar informatif saat melakukan feature selection dalam pembelajaran mesin. Dalam konteks ini, Gain Ratio bukan hanya digunakan dalam algoritma pohon keputusan, tetapi juga untuk prapemrosesan data pada algoritma lain seperti KNN, Naive Bayes, atau SVM.

Selain meningkatkan akurasi, proses seleksi fitur dengan Gain Ratio juga mengurangi kompleksitas model. Artinya, model dapat dilatih lebih cepat dan membutuhkan lebih sedikit sumber daya komputasi, karena hanya fitur-fitur penting yang digunakan. Ini sangat berguna ketika kita bekerja dengan dataset besar atau saat model akan dijalankan pada perangkat dengan daya terbatas.

Namun demikian, perlu dicatat bahwa jika nilai Split Information mendekati nol, maka nilai Gain Ratio bisa menjadi sangat besar dan menyesatkan. Oleh karena itu, biasanya dalam implementasi algoritma seperti C4.5, atribut dengan Information Gain yang sangat kecil akan diabaikan untuk mencegah bias akibat pembagian dengan angka yang hampir nol. Dalam implementasi di perangkat lunak seperti Weka atau Scikit-learn (meskipun lebih umum dalam ID3/C4.5), Gain Ratio dapat diakses sebagai salah satu metode seleksi atribut. Praktisi data science dapat memanfaatkan ini untuk menyaring fitur terbaik secara otomatis sebelum melakukan pelatihan model utama.

Secara keseluruhan, Gain Ratio merupakan penyempurna Information Gain yang membuat proses pemilihan atribut lebih adil dan tidak bias. Dengan memanfaatkan Gain Ratio dalam proses klasifikasi atau seleksi fitur, model yang dibangun akan lebih akurat, efisien, dan memiliki generalisasi yang lebih baik terhadap data baru.

II.4 Penelitian Terdahulu

Tabel II. 1 Tabel Penelitian Terkait

No	Peneliti (Tahun)	Metode	Hasil
1	(Ligthart et al., 2021)	*LSTM* dan *CNN*	Penelitian ini menunjukkan bahwa algoritma Naive Bayes masih menjadi salah satu metode klasifikasi yang paling efisien, terutama saat digunakan untuk analisis teks dan sentimen. Meskipun algoritma ini menggunakan pendekatan probabilistik yang sederhana dan mengasumsikan bahwa setiap fitur saling independent yang dalam praktiknya tidak selalu benar hasil yang dihasilkan tetap tergolong akurat. Karena itulah, Naive Bayes banyak dipilih dalam pengolahan data berskala besar yang membutuhkan kecepatan dan efisiensi dalam proses komputasinya.
2	(Yusra et al., 2021)	*KNN* dan *Relief-F*	Pada penelitian ini metode Relief-F merupakan teknik seleksi atribut yang efektif dalam mengidentifikasi fitur-fitur relevan dengan mempertimbangkan jarak antar instance dalam dataset. Metode

			ini memberikan bobot pada setiap atribut berdasarkan kemampuannya dalam membedakan data dari kelas yang berbeda, sekaligus menjaga kesamaan pada data dari kelas yang sama. Pendekatan ini sangat berguna dalam menangani dataset berdimensi tinggi, di mana tidak semua atribut memiliki kontribusi signifikan terhadap hasil klasifikasi. Dalam penelitian tersebut, kombinasi antara Relief-F dengan algoritma klasifikasi seperti K-Nearest Neighbor (KNN) dan Naive Bayes terbukti mampu meningkatkan akurasi model secara signifikan. Penggunaan Relief-F tidak hanya menyederhanakan jumlah fitur yang diproses, tetapi juga memperbaiki efisiensi dan ketepatan dalam prediksi, menjadikannya pilihan yang tepat dalam analisis data berskala besar
3	(Elİaçik, 2025)	*KNN, SVM, Random Forest, Naïve Bayes, Logistic*	Penelitian oleh Berat Eliaçik dan Ali Hakan Işık (2025) berfokus pada membandingkan efektivitas sejumlah algoritma klasifikasi

		Regression, Decission Tree	dalam memprediksi risiko penyakit jantung berdasarkan data medis yang mencakup 1190 sampel dengan 13 fitur utama, seperti usia, jenis kelamin, tekanan darah, dan kadar kolesterol. Dengan memanfaatkan bahasa pemrograman Python dan platform Jupyter Notebook, peneliti menguji algoritma seperti K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Random Forest, Naive Bayes, Logistic Regression, dan Decision Tree menggunakan ukuran kinerja seperti akurasi, presisi, sensitivitas, dan F1-score. Hasilnya menunjukkan bahwa KNN, SVM, dan Random Forest memberikan hasil paling optimal, masing-masing dengan akurasi 88,52% dan F1-score sebesar 86,79%, sementara Naive Bayes dan Logistic Regression tetap memberikan performa yang cukup baik. Sebaliknya, Decision Tree menunjukkan hasil yang kurang memuaskan dibanding algoritma lainnya. Secara keseluruhan, studi ini menyoroti

			pentingnya pemanfaatan kecerdasan buatan dan teknik data mining dalam mendukung diagnosis dini penyakit jantung, serta memberikan gambaran bagi pengembangan sistem pendukung keputusan di bidang kesehatan.
4	(Abdullah et al., 2025)	*KNN* dan *Relief-F*	Penelitian yang dilakukan oleh Anita, Asrul Abdullah, dan Syarifah Putri Agustini Alkadri (2025) bertujuan untuk mengklasifikasikan kondisi kesehatan janin dengan memanfaatkan algoritma K-Nearest Neighbor (KNN) yang dipadukan dengan metode seleksi fitur Relief-F. Mereka menggunakan data dari dataset Cardiotocography yang tersedia secara publik dan berisi lebih dari dua ribu rekaman dengan berbagai parameter penting seperti detak jantung janin dan kontraksi rahim. Fokus utama penelitian ini adalah untuk melihat bagaimana pemilihan fitur yang tepat dapat meningkatkan akurasi model klasifikasi. Setelah diuji,

			kombinasi KNN dan Relief-F mampu menghasilkan akurasi tinggi hingga 89,6%, dan tetap stabil meskipun beberapa fitur penting dihilangkan, dengan akurasi hanya turun sedikit menjadi 86,6%. Temuan ini menunjukkan bahwa Relief-F sangat membantu dalam menentukan fitur yang paling berpengaruh, sekaligus mempercepat proses pelatihan model. Secara keseluruhan, studi ini menegaskan bahwa pendekatan kombinasi ini tidak hanya meningkatkan ketepatan prediksi, tetapi juga memberikan solusi praktis dan efisien dalam mendukung pemantauan kesehatan janin
5	(Saelan et al., 2020)	*Naive Bayes* dan *Decission Tree*	Algoritma Naïve Bayes memiliki akurasi sebesar 82.60% sedangkan pada pengujian algoritma Decision Tree memiliki akurasi sebesar 86.44%. Dengan melihat statistik dari kurva ROC terdapat Equal Error Rate (EER) dari masing-masing algoritma, Naïve Bayes memiliki tingkat EER sebesar 65%, sedangkan Decision Tree

			memiliki tingkat EER sebesar 55%, berdasarkan tingkat EER dapat diartikan bahwa algoritma Naïve Bayes memiliki kinerja lebih buruk jika dibandingkan dengan algoritma Decision Tree yang memiliki tingkat EER lebih rendah.
6	(Mehbodniya et al., 2022)	*Naïve Bayes*	Penelitian ini menggunakan dataset CTG (Cardiotocography) dari UCI Machine Learning Repository untuk mengklasifikasikan kesehatan janin ke dalam tiga kelas: normal, suspect, dan pathologic. Dari berbagai algoritma yang diuji, Random Forest menunjukkan performa terbaik dengan akurasi sebesar 93%, diikuti oleh MLP (Multi-Layer Perceptron) dan SVM. Dataset terdiri dari 2126 sampel dengan 21 fitur. Penelitian ini menekankan pentingnya pemilihan model yang tepat untuk sistem pendukung keputusan medis berbasis machine learning
7	(Al Duhayyim et al., 2023)	*Random Forest* and *XGBoost*	Penelitian ini menggunakan metode ensemble learning (RF

			dan XGBoost) untuk klasifikasi data kesehatan janin dengan dataset CTG. RF memberikan akurasi 98%, sementara XGBoost mencapai 99%. Penelitian juga menggunakan teknik oversampling untuk menyeimbangkan kelas. Semua model diuji dengan presisi, recall, dan F1-score dan menunjukkan performa luar biasa dalam mendeteksi kategori suspect dan pathologic.
8	(Nazli et al., 2025)	*TabNet, RF, XGBoost*	Penelitian ini membandingkan model deep learning TabNet dengan ensemble learning pada data CTG. TabNet memperoleh akurasi sebesar 94,36%, sedangkan model stacking dengan RF dan XGBoost mencapai 95,9%. Penelitian ini menyoroti kemampuan TabNet dalam bekerja pada data tabular, serta potensi stacking model untuk meningkatkan akurasi klasifikasi.
9	(Jebadurai et al., 2022)	*Relief-F + NB, KNN, SVM, DT*	Penelitian ini mengevaluasi efektivitas tiga teknik seleksi fitur (ANOVA, ROC-AUC, dan Spearman) dan mengujinya

			terhadap berbagai algoritma. Hasil menunjukkan peningkatan akurasi sebesar 2–4% setelah penerapan fitur seleksi. Naive Bayes, KNN, dan SVM memperlihatkan peningkatan stabil pada presisi dan F1-score. Kombinasi fitur seleksi berbasis statistik dengan algoritma ringan sangat cocok untuk klasifikasi medis berskala besar
10	(Bhardwaj et al., 2025)	*Active Learning* + *KNN*	Menggunakan metode active learning untuk meningkatkan efisiensi pelatihan model KNN pada data CTG. Penelitian ini menunjukkan bahwa dengan hanya 20% data pelatihan, model sudah dapat mencapai akurasi 99,06%. Teknik ini sangat efisien untuk diterapkan di bidang kesehatan karena mengurangi kebutuhan data besar
11	(Chulde-Fernández et al., 2025)	*MLP* (Neural Network)	Penelitian ini mengembangkan model klasifikasi menggunakan MLP pada data penyakit jantung. MLP memberikan akurasi 94% dan F1-score sebesar 84%. Model ini unggul dalam memprediksi pasien dengan risiko tinggi, meskipun recall

			untuk kelas negatif sedikit lebih rendah yaitu 75%. Penelitian ini mendukung penerapan deep learning untuk diagnosa awal penyakit jantung.
12	(Mao et al., 2025)	*Logistic Regression, Random Forest*	Dalam penelitian ini, Logistic Regression menunjukkan akurasi tertinggi (92,1%) dibandingkan Random Forest dan KNN. Dataset berasal dari data medis kardiovaskular yang dikumpulkan dari Bangladesh. Penelitian ini menunjukkan keandalan model linier untuk prediksi penyakit jantung pada populasi Asia Selatan.
13	(Chintaluri A, 2025)	*KNN, Random Forest* dan *Neural Network*	Penelitian ini membandingkan KNN, Random Forest, dan Neural Network untuk klasifikasi penyakit jantung dengan dataset dari UCI. RF memberikan akurasi terbaik sebesar 86,6% dan AUC mencapai 0,981, sedangkan KNN memiliki akurasi 88,8% dengan AUC 0,93. Neural Network sedikit lebih rendah tetapi tetap kompetitif
14	(Chintaluri A, 2025)	*Logistic Regression,*	Penelitian ini melakukan perbandingan efektivitas beberapa algoritma machine

		Random Forest, SVM, dan *KNN*	learning dalam memprediksi penyakit jantung dengan menggunakan dataset UCI Heart Disease. Algoritma yang diuji meliputi Logistic Regression, Random Forest, Support Vector Machine (SVM), dan K-Nearest Neighbor (KNN). Peneliti juga mengevaluasi pengaruh seleksi fitur menggunakan metode Mutual Information dan Chi-square terhadap kinerja model. Hasil penelitian menunjukkan bahwa tanpa menggunakan seleksi fitur, algoritma Random Forest menghasilkan akurasi tertinggi sebesar 89,7%, diikuti oleh SVM dengan akurasi 87%, dan Logistic Regression dengan 84,2%. Namun, ketika fitur diseleksi menggunakan Mutual Information, akurasi Random Forest sedikit menurun menjadi 85,3%, sementara SVM tetap stabil di 87%, dan Logistic Regression menjadi 82,6%. Begitu pula dengan metode Chi-square, akurasi Random Forest dan Logistic Regression sama-sama menjadi 83,2%, sedangkan SVM tetap di 82,6%. Penelitian

			ini menyimpulkan bahwa seleksi fitur tidak selalu berdampak positif terhadap peningkatan akurasi dan bahwa Random Forest merupakan model yang paling konsisten kinerjanya baik dengan maupun tanpa seleksi fitur
15	(Hossain, 2024)	*Logistic Regression, KNN, RF, Naive Bayes,* dan *SVC*	penelitian komparatif terhadap beberapa algoritma klasifikasi machine learning untuk prediksi penyakit jantung berdasarkan data klinis. Algoritma yang diuji meliputi K-Nearest Neighbor (KNN), Logistic Regression, Random Forest, Naive Bayes, dan Support Vector Classifier (SVC). Evaluasi dilakukan berdasarkan metrik akurasi, presisi, recall, F1-score, dan G-Mean. Hasil penelitian menunjukkan bahwa algoritma KNN memberikan performa terbaik dengan akurasi mencapai 92%, mengungguli semua algoritma lainnya dalam studi ini. Fitur-fitur klinis yang memiliki kontribusi signifikan terhadap akurasi prediksi meliputi jenis nyeri dada (chest pain type) dan detak jantung maksimal.

			Penelitian ini menyimpulkan bahwa dengan persiapan data klinis yang baik dan fitur yang relevan, algoritma KNN dapat menjadi pilihan andal dalam pengembangan sistem prediksi penyakit jantung berbasis machine learning.

II.5 Kontribusi Penelitian

Berdasarkan hasil studi pustaka dan ringkasan penelitian terdahulu, terdapat beberapa celah penelitian (*research gap*) yang melatarbelakangi dilakukannya penelitian ini. Pertama, sebagian besar penelitian sebelumnya hanya berfokus pada penggunaan satu metode seleksi atribut atau membandingkan algoritma klasifikasi tanpa secara spesifik mengevaluasi dampak seleksi atribut terhadap kinerja algoritma Naive Bayes. Misalnya, penelitian Yusra et al. (2021) dan Abdullah et al. (2025) menggunakan metode Relief-F dengan algoritma KNN, namun tidak membandingkannya dengan metode seleksi atribut lain seperti Gain Ratio, serta tidak menguji penerapannya secara spesifik pada Naive Bayes. Kedua, penelitian yang melibatkan Naive Bayes umumnya hanya menjadikan algoritma ini sebagai pembanding, bukan sebagai objek utama evaluasi performa, seperti terlihat pada studi Elİaçik (2025) yang lebih menekankan pada performa algoritma KNN, SVM, dan Random Forest. Selain itu, sebagian besar studi tersebut hanya menggunakan satu jenis dataset dan belum menguji efektivitas metode seleksi atribut dalam konteks data yang kompleks dan berskala besar. Kontribusi utama dari penelitian ini adalah menghadirkan analisis komparatif antara dua metode seleksi atribut populer, yaitu Relief-F dan Gain Ratio, dalam meningkatkan akurasi klasifikasi Naive Bayes. Penelitian ini secara eksplisit menempatkan Naive Bayes sebagai objek utama yang diuji dengan dan tanpa seleksi atribut, sehingga dapat memberikan gambaran empiris yang lebih mendalam tentang pengaruh masing-masing metode terhadap performa klasifikasi. Selain itu, penelitian ini

menggunakan dua dataset yang berbeda secara karakteristik, yaitu *House Vote* yang bersifat simbolik dan berukuran sedang, serta Bank Marketing yang berukuran besar dan lebih kompleks. Pendekatan ini memungkinkan peneliti untuk mengamati bagaimana perilaku metode seleksi atribut berubah dalam konteks data yang bervariasi. Hasil dari penelitian ini diharapkan dapat menjadi referensi yang berguna bagi peneliti dan praktisi dalam memilih strategi seleksi atribut yang sesuai untuk meningkatkan efisiensi dan efektivitas model klasifikasi, khususnya pada algoritma Naive Bayes.