# BAB II
# LITERATURE REVIEW

## 2.1 Kajian Pustaka

**Tabel 2. 1 Resume dari Kelima Penelitian**

| Riset | Tujuan | Pendekatan | Hasil |
|---|---|---|---|
| Dadang Kurniawan , Rizal Maulana , Mochammad Hannats Hanafi Ichsan, April 2019 | sistem otomatisasi untuk mendeteksi tingkat keparahan penyakit paru-paru dari pasien, sehingga ketika seorang pasien datang untuk memeriksakan paru-paru | Menggunakan metode *Naive Bayes*, sebagai algoritma yang mempunyai akurasi yang baik dan dapat digunakan berdasarkan penggolongan kelas diawal proses | Implementasi pendeteksi penyakit paru-paru berdasarkan warna kuku dan suhu tubuh berbasis sensor TCS3200 dan sensor LM35 dengan metode Naive Bayes. Mendapatkan hasil data dari 24 data yang diuji sebayak 12 data, diperoleh akurasi sebesar 91,6%. |
| Jefri Junifer Pangaribuan, Henry Tanjaya , Kenichi, Juli 2021 | Mendeteksi penyakit jantung dengan menggunakan data pasien. Terdapat berbagai jenis metode yang dapat digunakan untuk mendiagnosis apakah seseorang | Penelitian ini implementasi penggunaan algoritma yaitu Regresi logistik, dimana algoritma tersebut memakai fungsi logistik untuk menghasilkan binary atau nol dan satu sebagai | Pada data training, metode Regresi Logistik mempunyai nilai sensitivity yang paling tinggi yaitu 88.54% dibanding metode lainnya. Pada data testing, metode Regresi Logistik mempunyai nilai kekhususan yang paling tinggi yaitu 87.50% |

| Riset | Tujuan | Pendekatan | Hasil |
|---|---|---|---|
| | terkena penyakit jantung atau tidak. | penentuan klasifikasi | dibanding metode lainnya. |
| Fata Nidaul Khasanah, Desember 2016 | Membandingkan hasil yang diperoleh dari empat algoritma, yaitu J48, Naïve Bayes, *OneR* dan ZeroR. | Teknik pengelompokan data yang didapat dari hasil mining, dengan menggunakan empat metode algoritma yaitu J48, Naïve Bayes, *OneR* dan ZeroR. | Algoritma *Naive Bayes* merupakan algoritma yang mempunyai nilai akurasi tertinggi yaitu 96.74%, selanjutnya adalah algoritma J48 dengan 93.48%, algoritma *OneR* 90.22%, dan algoritma ZeroR merupakan algoritma yang mempunyai nilai akurasi terendah yaitu 59.78% |
| Maria ArtatiEka Setyorin, March 2018 | Untuk menganalisis metode *Random Forest* dalam mendeteksi kangker paru-paru | Penelitian ini tentang deteksi kanker menggunakan algoritma *data mining* dengan metode *machine learning* Random Forest dan s*upport vector machine* (SVM). | Hasil klasifikasi Random Forest menghasilkan akurasi sebesar 90,32%. Sedangkan, hasil klasifikasi SVM menghasilkan akurasi sebesar 87,10%. |

| Riset | Tujuan | Pendekatan | Hasil |
|---|---|---|---|
| I Ketut Agung Enriko, June2019 | Untuk mendiagnosis penyakit jantung sebagai penyebab utama kematian selama bertahun-tahun | Dalam mendeteksi penyakit jantung menggunakan metode *data mining*, dengan menggunakan algoritma yaitu K-Nearst Neighbor,*CART* dan *AdaBoost* | Penelitian c*omparative study of heartdisease diagnosis using top ten Data mining classification agorithms,* mendapatkan hasil dari ketiga algoritma tersebut yaitu *Random Tree* (78,0%),KNN(71,6%), dan MLP (63%). |

## 2.2 Dasar Teori

### 2.2.1 Pengolahan Citra

Citra dapat dibagi menjadi dua jenis yaitu citra diam (*still images*) dan citra bergerak (*moving images*). Citra diam adalah citra tunggal yang tidak ditampilkan secara beruntun, sedangkan citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun atau berurutan dalam interval waktu yang singkat sehingga memberi kesan pada mata seolah-olah bergerak, misalnya dalam animasi. Dalam penelitian ini selanjutnya citra diam disebut sebagai citra. Menurut cara pembentukannya citra dibedakan menjadi citra kontinyu dan citra diskrit. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog, misainya mata manusia atau kamera analog. Citra diskrit dihasilkan dari proses digitalisasi terhadap citra kontinu. Terdapat sistem yang memungkinkan fungsi digitalisasi sehingga tercipta citra diskrit, diantaranya adalah pemindai (*scanner*) atau kamera digital. Citra diskrit lebih sering disebut dengan citra digital [3].

Citra adalah sebuah fungsi intensitas cahaya 2 dimensi $f_{(x,y)}$ dimana parameter x adalah posisi baris dan parameter y adalah posisi kolom, sedangkan f adalah intensitas atau kecerahan dari citra pada koordinat (x, y). Istilah citra sering

disebut juga dengan citra digital atau gambar digital (*digital image*). Istilah citra digital tersebut sering digunakan untuk menjelaskan bahwa citra tersebut sudah diubah menjadi citra secara digital sehingga citra tersebut dapat diproses dengan komputer. Setiap titik dalam citra mempunyai intensitas warna yang disebut derajat keabuan. Pada umumnya citra berbentuk empat persegi panjang dan dimensi ukurannya dinyatakan sebagai tinggi x lebar [3].

Pengolahan citra (*image processing*) adalah memproses citra khususnya dengan menggunakan komputer menjadi citra yang kualitasnya lebih baik. Pada umumnya operasi pada pengolahan citra diterapkan pada citra apabila:

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokan, dicocokkan atau diukur.
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

Pada prinsipnya pengolahan citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia atau mesin (dalam hal ini komputer). Teknik – teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukkannya adalah citra maka keluarannya juga citra. Citra keluaran tersebut mempunyai kualitas citra yang lebih baik daripada citra masukan. Citra yang kualitasnya tidak seperti yang diinginkan. Misalnya citra yang terlalu gelap atau terlalu terang, bahkan citra yang kabur, warna yang kurang tajam dan sebagainya. Untuk memanipulasi citra yang semacam ini maka diperlukan teknik mengolah citra tersebut. Teknik mengolah citra dinamakan pengolahan citra [3] [13].

### 2.2.2 Penyakit Paru-Paru

Penyakit paru-paru adalah kelompok kondisi yang mempengaruhi organ paru-paru, yang memiliki peran penting dalam proses pernapasan dan pertukaran oksigen dalam tubuh manusia. Penyakit paru-paru dapat disebabkan oleh berbagai faktor, termasuk infeksi, paparan bahan kimia berbahaya, merokok, dan faktor genetik. Pemahaman dasar tentang penyakit paru-paru meliputi struktur dan fungsi paru-paru, patogenesis penyakit, serta faktor risiko dan gejala yang terkait.

Paru-paru adalah organ vital dalam sistem pernapasan yang terdiri dari saluran bronkial dan jaringan paru yang mengandung jutaan gelembung udara kecil yang disebut alveoli. Fungsi utama paru-paru adalah mengambil oksigen dari udara yang dihirup dan mengeluarkan karbon dioksida sebagai produk sampingan metabolisme. Paru-paru juga memiliki mekanisme pertahanan untuk melawan patogen dan partikel berbahaya yang masuk ke dalam saluran napas.

Patogenesis penyakit paru-paru melibatkan berbagai mekanisme yang dapat mempengaruhi fungsi normal paru-paru. Proses inflamasi adalah respons umum yang terjadi dalam banyak penyakit paru-paru. Inflamasi kronis dapat mengakibatkan kerusakan jaringan paru-paru dan perubahan struktural yang mengganggu aliran udara dan pertukaran gas. Selain itu, faktor genetik juga dapat memengaruhi kerentanan seseorang terhadap penyakit paru-paru tertentu, seperti asma atau fibrosis paru.

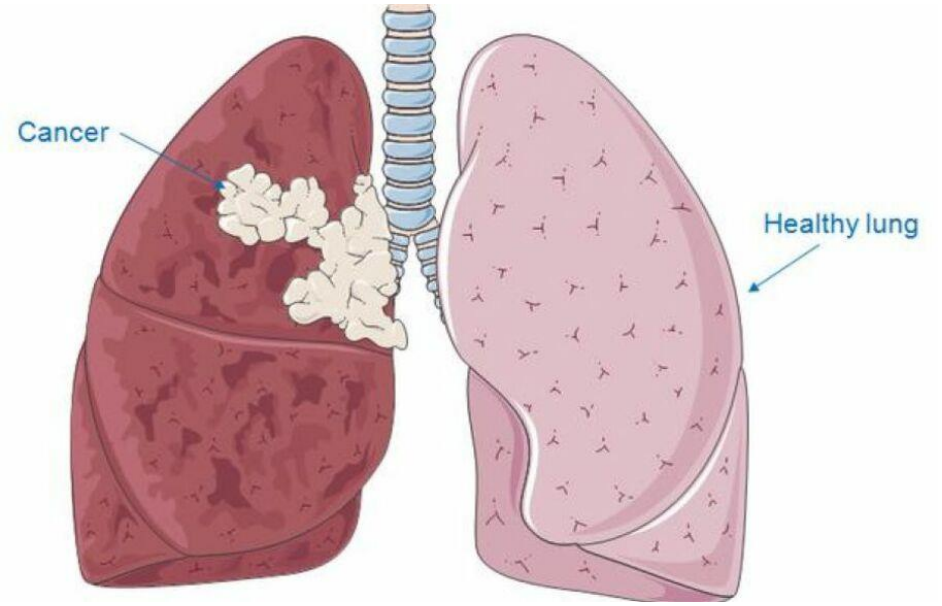
Faktor risiko yang berhubungan dengan penyakit paru-paru dapat bervariasi tergantung pada jenis penyakitnya. Merokok tembakau adalah faktor risiko utama untuk penyakit paru-paru, termasuk kanker paru-paru dan penyakit paru obstruktif kronik (PPOK). Paparan terhadap polusi udara, asap rokok, serbuk kayu, serat asbestos, bahan kimia berbahaya, dan infeksi saluran napas juga dapat berkontribusi terhadap perkembangan penyakit paru-paru.

Gejala penyakit paru-paru juga dapat bervariasi tergantung pada jenis dan tingkat keparahannya. Gejala umum meliputi batuk kronis, sesak napas, nyeri dada, produksi dahak yang berlebihan, kelelahan, dan penurunan berat badan yang tidak dijelaskan. Gejala ini dapat memengaruhi kualitas hidup seseorang dan memerlukan intervensi medis yang tepat.

Pengelolaan penyakit paru-paru melibatkan berbagai pendekatan tergantung pada jenis dan tingkat keparahannya. Pengobatan dapat mencakup penggunaan obat-obatan seperti bronkodilator, kortikosteroid, antibiotik, dan terapi oksigen. Pada kasus yang lebih parah, transplantasi paru-paru mungkin diperlukan. Selain itu, perubahan gaya hidup seperti ber berhenti merokok, menjaga pola makan sehat, dan melakukan olahraga secara teratur juga merupakan bagian dari pengelolaan penyakit paru-paru. Rehabilitasi paru juga dapat membantu meningkatkan fungsi paru-paru dan kualitas hidup pasien. Pencegahan juga memiliki peran penting

dalam mengurangi beban penyakit paru-paru. Vaksinasi seperti vaksin influenza dan vaksin pneumonia direkomendasikan untuk melindungi terhadap infeksi yang dapat memperburuk kondisi paru-paru. Selain itu, menghindari paparan terhadap polusi udara, bahan kimia berbahaya, dan asap rokok juga sangat penting dalam pencegahan penyakit paru-paru.

Dalam hal penelitian, studi tentang penyakit paru-paru terus dilakukan untuk memperdalam pemahaman kita tentang penyebab, faktor risiko, mekanisme patogenesis, dan pengobatan yang lebih efektif. Penelitian ini melibatkan pengembangan teknik diagnostik yang lebih sensitif dan spesifik, pengenalan terapi yang lebih efektif dan inovatif, serta identifikasi faktor risiko yang dapat diubah untuk mencegah penyakit paru-paru.




**Gambar 2. 1 Tampilan Kondisi Paru-Paru Sehat dan Penyakit Paru-Paru [18]**

### 2.2.3 *Naïve Bayes*

Naive Bayes adalah pengklasifikasi statistik yang dapat digunakan untuk memprediksi keanggotaan kelas. Naive Bayes didasarkan pada teorema Bayes yang memiliki kemampuan klasifikasi mirip dengan pohon keputusan dan saraf jaringan. Naive Bayes terbukti memiliki akurasi dan kecepatan tinggi ketika diterapkan ke database dengan data yang besar [5]. Contoh dari penerapan algoritma Naive Bayes classification pengklasifikasian penerimaan mahasiswa

baru dengan menerapkan metode probabilitas dan stastistik, yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya.

Konsep dasar dari Naïve Bayes adalah menghitung probabilitas kelas berdasarkan probabilitas atribut atau fitur yang ada dalam data. Dalam kasus klasifikasi biner (dua kelas), seperti kasus "penyakit paru-paru" atau "tidak penyakit paru-paru," algoritma Naïve Bayes menghitung probabilitas untuk setiap kelas berdasarkan atribut yang terdapat pada data. Selanjutnya, algoritma akan memilih kelas dengan probabilitas tertinggi sebagai hasil prediksi untuk data yang diberikan.

Meskipun algoritma Naïve Bayes dianggap "naif" karena mengasumsikan bahwa atribut atau fitur dalam data adalah independen satu sama lain (tidak saling mempengaruhi), algoritma ini tetap memberikan hasil yang cukup baik dalam banyak kasus, terutama pada data yang besar dan berdimensi tinggi.

Proses algoritma Naïve Bayes dapat dijelaskan sebagai berikut:

1. Hitung probabilitas prior untuk setiap kelas berdasarkan jumlah data dalam setiap kelas dibagi dengan total data keseluruhan.
2. Hitung probabilitas kondisional untuk setiap atribut dalam setiap kelas berdasarkan jumlah kemunculan atribut tersebut dalam kelas tertentu dibagi dengan total kemunculan atribut di seluruh data.
3. Hitung probabilitas posterior untuk setiap kelas berdasarkan hasil perhitungan probabilitas prior dan probabilitas kondisional.
4. Pilih kelas dengan probabilitas posterior tertinggi sebagai hasil prediksi untuk data yang diberikan.

Contoh penerapan Naïve Bayes dapat ditemukan dalam berbagai bidang, seperti klasifikasi spam email, klasifikasi teks berita, atau prediksi penyakit berdasarkan gejala dan parameter kesehatan. Meskipun Naïve Bayes cukup sederhana, ia sering memberikan hasil yang memuaskan dalam berbagai kasus dan dapat menjadi pilihan algoritma yang baik untuk tugas klasifikasi dengan data yang besar dan berdimensi tinggi.

Prediksi Naive Bayes yang didasarkan oleh formula teorema Bayes dengan formula sebagai berikut :

$$P(H|X) = \frac{P(X|H)\, xP(H)}{P(X)} \quad \ldots. (1)$$

X : Data dengan class yang belum diketahui

H : Hipotesis data merupakan suatu class spesifik

P(H|X) : Probabilitas hipotesis H berdasar kondisi X (posteriori probabilitas) P(H) : Probabilitas hipotesis H (prior probabilitas)

P(X|H) : Probabilitas X berdasarkan kondisi pada hipotesis H P(X) : Probabilitas X

### 2.2.4 *Regresi Logistik*

Regresi Logistik adalah analisis pendekatan model matematis yang digunakan untuk menganalisis hubungan satu atau beberapa variabel independen dengan sebuah variabel dependen (dua kemungkinan). Regresi Logistik digunakan untuk mendeskripsikan data dan menjelaskan hubungan antara satu variabel biner dependen dan satu atau lebih variabel bebas nominal, ordinal, interval atau rasio tingkat.

Model Regresi logistik adalah model statistik yang digunakan untuk mengetahui pengaruh variabel prediktor (X) terhadap variabel respon (Y) dengan variabel respon adalah data dikotomi yang bernilai 1 berarti variabel respon memiliki kriteria bahwa ditentukan dan 0 menunjukkan bahwa variabel respon tidak memiliki kriteria yang ditentukan [6]. Contoh penerapan Regresi logistik dapat memprediksi mahasiswa statistika lulus tepat waktu, ketetapan klasifikasi atau tingkat akurasi model Regresi Logistik dalam mengklasifikasi mahasiswa lulus sebesar 45.69%.

Tujuan utama dari Regresi Logistik adalah untuk memodelkan hubungan antara variabel prediktor dengan probabilitas terjadinya suatu kejadian pada variabel respon. Dalam konteks klasifikasi biner, Regresi Logistik digunakan untuk memprediksi probabilitas suatu kejadian berdasarkan nilai-nilai variabel prediktor. Jika probabilitas tersebut melebihi nilai ambang tertentu (misalnya 0.5), maka kejadian tersebut akan diklasifikasikan sebagai kategori positif, sedangkan jika probabilitasnya kurang dari nilai ambang, maka akan diklasifikasikan sebagai kategori negatif.

Proses Regresi Logistik dapat dijelaskan sebagai berikut:

1. Melakukan transformasi variabel dependen: Jika variabel respon adalah biner (misalnya ya/tidak), maka dapat digunakan transformasi seperti

logit atau sigmoid untuk mengubah variabel biner menjadi skala kontinu, sehingga dapat diaplikasikan pada model regresi.

2. Menyesuaikan model regresi: Model regresi logistik digunakan untuk mengestimasi parameter-parameter yang mempengaruhi variabel respon berdasarkan variabel prediktor. Ini melibatkan proses mencari garis regresi yang paling sesuai untuk mendekati hubungan antara variabel prediktor dan probabilitas kejadian.
3. Evaluasi model: Setelah model regresi logistik diperoleh, model tersebut diuji dengan menggunakan data yang tidak digunakan dalam proses pelatihan. Evaluasi model dilakukan untuk memeriksa sejauh mana model tersebut mampu memprediksi kejadian dengan akurat.

Persamaan model Regresi :

$$g(x_i) = \frac{e^{\beta_0 + \beta_1 + x_{1i}}}{1 + e^{\beta_0 + \beta_1 + x_{1i}}} \dots (2)$$

$e^{\beta_0}$ = jumlah parameter

$(x_i)$ = *predictor*

Dengan fungsi logit $g(xi)$ yaitu:

$$g(x_i) = \ln\left[\frac{\pi(x_i)}{1-\pi(x_i)}\right] = \beta_0 + \beta_{1x_{1i}} \dots (3)$$

$(xi)$ = *predictor*

$e\ \beta_0$ =jumlah parameter

$g(xi)$ = model logit

Dari persamaan (1) dan (2) dapat disederhanakan menjadi:

$$\pi(x_i) = \frac{e^{g(x_i)}}{1 + e^{g(x_i)}} \dots (4)$$

$x_i$ = *predictor*

$e^{g(x_i)}$ = penyederhanaan dari fugsi logit dan model regresi

### 2.2.5 *K-Nearest Neighbor*

Algoritma *K-Nearest Neighbor* (KNN) merupakan salah satu metode klasifikasi dalam *data mining*, dimana KNN mengklasifikasikan sekumpulan data

berdasarkan data pembelajaran yang telah diklasifikasikan atau diberi label. KNN termasuk dalam *supervied learning group* yaitu hasil dari *instancequery* yang baru diklasifikasikan berdasarkan mayoritas kedekatan dengan kategori yang ada di KNN [7].

*K-Nearst Neigbor* (KNN) mempuyai prinsip membandingkan *data testing* ( data baru ) dengan *data training* ( data lama ) secara satu persatu. KNN memiliki bebrapa kelebihan yaitu ketangguhan terhadap *training data* KNN yang memiliki banyak *noise* dan efektif apabila *training data* yang besar dan sedangkan kelemahanya KNN perlu menentukan nilai dari parameter *training* berdasarkan jarak [8]. Contoh penerapan metode KNN untuk sistem rekomendasi pemilihan mobil metode KNN merupakan suatu bentuk model pendukung keputusan yang dapat mengklasifikasi data bedasarkan jarak terdekat. Metode ini dirancang untuk membantu calon pembeli dalam memilih mobil berdasarakan tujuan pembelian berupa mobil untuk bisnis, mobil keluarga, dan mobil angkutan barang, harga, tahun pembuatan, kapasitaspenumpang, warna, kapasitas mesin, jenis transmisi. Rumus KNN dituliskan pada persamaan di bawah ini:

Yaitu:

$$d(x, y) = \sqrt{\sum_{i=1}^{a}(x_1 - y_i)^2} \qquad \ldots\ldots (5)$$

Keterangan:

D adalah jarak *Euclidean*

X adalah koordinat data training

Y adalah koordinat data uji

Proses algoritma KNN dapat dijelaskan sebagai berikut:

1. Menghitung jarak: Algoritma KNN menggunakan metrik jarak, seperti Euclidean distance, Manhattan distance, atau Minkowski distance, untuk mengukur jarak antara data yang akan diklasifikasikan dengan data lain dalam dataset.
2. Menentukan tetangga: KNN memilih k data terdekat (k tetangga terdekat) berdasarkan jarak yang dihitung sebelumnya. Nilai k biasanya ditentukan sebelumnya oleh pengguna.
3. Memilih mayoritas kelas: Setelah menentukan k tetangga terdekat, algoritma KNN akan memeriksa kelas dari tetangga-tetangga tersebut.

Hasil prediksi akan ditentukan berdasarkan mayoritas kelas dari tetangga-tetangga tersebut. Misalnya, jika lebih banyak tetangga termasuk ke dalam kelas A, maka data yang akan diklasifikasikan akan diprediksi sebagai anggota kelas A.

4. Algoritma KNN adalah algoritma yang non-parametrik, artinya algoritma ini tidak mengasumsikan distribusi data tertentu. KNN cukup sederhana dan mudah diimplementasikan, namun kecepatan komputasinya bisa menjadi tantangan pada dataset besar karena harus menghitung jarak dengan seluruh data dalam dataset.

KNN dapat digunakan untuk berbagai tugas, termasuk klasifikasi data, regresi, dan penyelesaian masalah clustering. Penggunaan nilai k yang tepat menjadi hal penting dalam algoritma ini, karena nilai k yang terlalu kecil dapat menyebabkan hasil yang tidak stabil atau overfitting, sedangkan nilai k yang terlalu besar dapat menyebabkan underfitting. Oleh karena itu, pemilihan nilai k yang optimal adalah salah satu pertimbangan penting dalam penerapan KNN.

### 2.2.6 *K-Star*

*Kstar* adalah pengklasifikasi berbasis contoh, yaitu kelas sampel uji didasarkan pada contoh kelas mereka dari latihan serupa, sebagaimana ditentukan oleh beberapa fungsi persamaan [9].

*Kstar* mempuyai konsep entropi untuk mendefinisikan metrik jarak yang dihitung dengan *mean*. Klasifikasi dengan *Kstar* dibuat dengan menjumlahkan probabilitas dari *instance* semua anggota dari sebuah kategori. Ini harus dilakukan dengan sisa kategori, untuk akhirnya memilih yang tertinggi kemungkinanya, dan untuk menangani nilai yang hilang dalam kumpulan data mengansumsikan bahwa probabilitas tranformasi itu ke jenis nilai, adalah rata-rata dari probabilitas [10]. Contoh penerapanya untuk ektrasi fitur tekstur dan warna pada kulit katak menggunakan glcm dan momen warna dengan klasifikasi dengan metode *Kstar*, dalam penelitian ini menggunakan data ekstrasi tekstur dan warna untuk menentukan akurasi jenis katak.Rumus *Kstar* ditulis dalam persamaan di bawah ini:

$$K*(b|a) = -log_2 P*(b|a) \ldots.(6)$$

Dimana P adalah fungsi probabilitas yang didefinisikan sebagai probabilitas semua

jalur dari *instance* a ke *instance* b.

### 2.2.7 *AdaBoost*

Algoritma *Adaptive Boosting* atau yang biasa disebut *AdaBoost* adala salah satu algoritma yang digunakan untuk pengambilan keputusan. Fokus dari metode ini adalah untuk menghasilkan serangkaian *base classifiers*. *Training set* yang digunakan untuk setiap *base classifier* dipilih berdasarkan performansi dari *classifier* sebelumnya. Di dalam *boosting*, sampel yang tidak diprediksikan dengan benar oleh *classifier* di dalam rangkaian akan dipilih lebih sering dibandingkan dengan sampel yang telah diprediksikan dengan benar. Dengan demikian, *AdaBoost* mencoba menghasilkan *base classifier* baru yang lebih baik untuk memprediksikan sampel pada *base classifier* sebelumnya memiliki performansi yang buruk [11].

Adaboost adalah salah satu metode *boosting*, yang berarti itu berfokus pada penggabungan model berurutan, dan model yang dihasilkan selanjutnya akan diberi bobot lebih pada data yang sebelumnya diklasifikasikan dengan kesalahan oleh model sebelumnya.

Proses algoritma Adaboost dapat dijelaskan sebagai berikut:

1. Inisialisasi Bobot: Pada awalnya, setiap sampel data dalam dataset diberi bobot yang sama. Bobot ini digunakan untuk mengatur pentingnya setiap sampel dalam proses pembuatan model.
2. Membangun Model Lemah: Adaboost memulai dengan membangun model klasifikasi lemah, seperti *Decision Stump* (model pohon keputusan yang hanya memiliki satu level) atau *Single-Feature Classifier*. Model lemah ini berfokus pada satu atribut (fitur) dan memiliki kemampuan prediksi yang terbatas.
3. Evaluasi dan Pemberian Bobot: Setelah model lemah dibangun, algoritma Adaboost mengevaluasi performa model pada dataset. Setiap sampel data yang diklasifikasikan dengan benar oleh model lemah akan diberi bobot lebih rendah, sementara sampel yang diklasifikasikan dengan kesalahan akan diberi bobot lebih tinggi. Bobot ini mencerminkan seberapa sulitnya model lemah untuk mengklasifikasikan

sampel tertentu.

4. Pemilihan Model Selanjutnya: Model lemah selanjutnya akan dibangun dengan memberi bobot lebih pada data yang diklasifikasikan dengan kesalahan oleh model sebelumnya. Hal ini berarti model selanjutnya akan lebih fokus untuk mengatasi sampel yang sebelumnya sulit diklasifikasikan.
5. Proses Iteratif: Langkah evaluasi, pemberian bobot, dan pemilihan model selanjutnya diulang beberapa kali (iterasi). Pada setiap iterasi, bobot sampel akan disesuaikan dan model lemah baru akan dibangun.
6. Gabungan Model: Setelah sejumlah model lemah dibangun, hasil prediksi dari semua model akan digabungkan dengan memberikan bobot pada masing-masing model. Hasil prediksi akhir akan didasarkan pada mayoritas suara dari model lemah yang berkontribusi.

Keunggulan Adaboost:

1. Meningkatkan Kinerja: *Adaboost* dapat meningkatkan kinerja model klasifikasi dengan menggabungkan model lemah menjadi model yang lebih kuat.
2. Ketahanan terhadap *Overfitting: Adaboost* memiliki kemampuan untuk mengurangi overfitting karena model lemah yang digunakan cenderung sederhana dan tidak kompleks.
3. Fleksibilitas: Algoritma *Adaboost* dapat digunakan dengan berbagai model lemah dan tidak terbatas pada model tertentu.

Kelemahan Adaboost:

1. Sensitif terhadap Data Noise: Adaboost dapat terpengaruh oleh data noise atau *outlier*, karena model lemah lebih mungkin untuk memberikan bobot lebih pada sampel yang sulit diklasifikasikan dengan benar.
2. Komputasi yang Mahal: Proses iteratif dalam Adaboost dapat memerlukan waktu komputasi yang lebih tinggi, terutama pada dataset yang besar atau dengan banyak model lemah.

Pemilihan *Adaboost* sebagai algoritma klasifikasi harus mempertimbangkan karakteristik dataset, kompleksitas model lemah, dan waktu komputasi yang tersedia. Dengan memilih model lemah yang sesuai dan mengoptimalkan

parameter *Adaboost,* algoritma ini dapat memberikan hasil prediksi yang akurat dan stabil.

Contoh penerapan algoritma *AdaBoost* adalah menentukan pola masuk calon mahasiswa dengan klasifikasi *AdaBoost* dalam mencari faktor-faktor yang membuat mahasiswa melanjutkan proses registrasi ulang.Gambar 2.1 di bawah menjelaskan *pseudo-code AdaBoost*.

**Input:** Data set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$;

Base learning algorithm $\mathcal{L}$;

Number of learning rounds $T$.

**Process:**

$D_1(i) = 1/m.$ % Initialize the weight distribution

for $t = 1, \cdots, T$:

$h_t = \mathcal{L}(\mathcal{D}, D_t)$; % Train a weak learner $h_t$ from $\mathcal{D}$ using distribution $D_t$

$\epsilon_t = \Pr_{i\sim D_t}[h_t(x_i \neq y_i)]$; % Measure the error of $h_t$

$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; % Determine the weight of $h_t$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(x_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(x_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$ % Update the distribution, where $Z_t$ is

% a normalization factor which enables $D_{t+1}$ be a distribution

end.

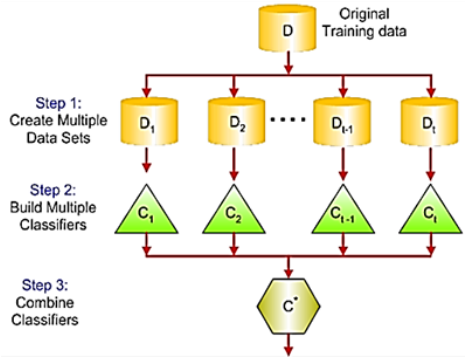
**Output:** $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

**Gambar 2. 2 *Pseudo-Code AdaBoost***

### 2.2.8 *Bagging*

*Bagging* adalah algoritma yang tepat untuk mengurangi *data noise* pada jaringan saraf, dan diterapkan dengan baik pada kumpulan data kuat yangmemiliki atribut dan label Numerik. *Bagging* adalah singkatan dari *bootstrap aggregating*, menggunakan sub-*dataset* (*bootstrap*) untuk menghasilkan *set* pelatihan L (*learning*), L melatih dasar belajar menggunakan prosedur pembelajaran yang tidak stabil, dan kemudian, selama pengujian mengambil rata-rata . *Bagging* baik digunakan untuk klasifikasi dan regresi. Dalam kasus regresi, untuk menjadi lebih kuat, seseorang dapat mengambil rata-rata ketika menggabungkan prediksi. *Bagging* adalah sebuah algoritma pembelajaran yang stabil pada perubahan kecil dalam *training set* menyebabkan perbedaan besar dalam peserta didik yang dihasilkan, yaitu algoritma belajar pada data yang memilikivarian tinggi (*noise*).

*Bagging* mampu meningkatkan akurasi secara signifikan lebih besar dibanding model individual, dan lebih kuat terhadap efek *noise* dan *overfitting* dari data pelatihan asli [12]. Contoh penerapan *Bagging* untuk memperbaiki hasil prediksi nasabah perusahaan asuransi untuk mengurangi tingkat kesalahan klasifikasi *cart*.Gambar 2.3 menggambarkan bagaimana algoritma *Bagging* bekerja.




**Gambar 2. 3 *Algoritma Bagging Menggambungkan Beberapa Model* [20]**

### 2.2.9 *OneR*

Algoritma *OneR* merupakan singkatan dari *One Rule*. Algoritma *One*Rakan membangkitkan sebuah aturan untuk setiap *attribute* kemudian memilih aturan dengan *error* paling kecil yang selanjutnya digunakan sebagai *One Rule*. Untuk membuat aturan (*rule*) setiap *attribute* yang ada maka perlu membuat tabel kemunculan untuk setiap *attribute* dengan targetnya [13].

One R (One Rule) adalah salah satu algoritma klasifikasi sederhana yang digunakan untuk menghasilkan aturan sederhana berdasarkan satu fitur (atribut) dalam dataset. Tujuan dari algoritma One R adalah untuk mencari aturan yang paling relevan dan efektif dalam mengklasifikasikan data.

Proses algoritma One R dapat dijelaskan sebagai berikut:

1. Seleksi Fitur: Algoritma One R memilih satu fitur (atribut) dari dataset yang akan digunakan sebagai dasar untuk membuat aturan klasifikasi. Fitur ini dipilih berdasarkan atribut yang paling informatif atau memiliki korelasi yang tinggi dengan kelas target.
2. Membuat Aturan: Setelah fitur terpilih, algoritma One R membuat aturan klasifikasi berdasarkan nilai-nilai unik dari fitur tersebut. Aturan ini bersifat sederhana dan berbentuk "Jika fitur adalah nilai X, maka kelas target adalah Y." Pada dasarnya, aturan ini mencoba untuk

mengklasifikasikan data berdasarkan frekuensi kemunculan nilai tertentu dari fitur yang dipilih.

3. Evaluasi Aturan: Setelah aturan dibuat, algoritma One R mengukur performa aturan tersebut pada dataset dengan menghitung tingkat akurasi atau tingkat kesalahan. Evaluasi dilakukan dengan menggunakan metrik seperti akurasi, presisi, recall, atau F1-score.
4. Seleksi Aturan Terbaik: Algoritma One R mencoba beberapa fitur sebagai dasar untuk aturan klasifikasi dan memilih aturan yang memberikan hasil prediksi terbaik (tingkat akurasi tertinggi) pada dataset.

Keunggulan One R:

1. Sederhana dan Mudah Diinterpretasi: Aturan yang dihasilkan oleh One R bersifat sederhana dan mudah diinterpretasi, sehingga mudah untuk dimengerti oleh manusia.
2. Cepat dan Efisien: Proses algoritma One R relatif cepat dan efisien karena hanya melibatkan satu fitur sebagai dasar untuk membuat aturan.
3. Cocok untuk Data Categorical: One R cocok untuk dataset yang berisi atribut kategorikal (nominal atau ordinal), karena aturan yang dihasilkan didasarkan pada nilai-nilai unik dari fitur tersebut.

Kelemahan One R:

1. Tidak Robust terhadap Atribut Lain: Algoritma One R hanya mempertimbangkan satu fitur untuk membuat aturan klasifikasi, sehingga dapat mengabaikan informasi penting dari atribut lain yang mungkin mempengaruhi klasifikasi.
2. *Overfitting*: *One R* cenderung menghasilkan aturan yang sangat spesifik untuk dataset pelatihan, yang dapat menyebabkan *overfitting* pada data tersebut dan hasil yang tidak optimal pada data baru.

*OneR* adalah algoritma klasifikasi sederhana yang membangun pohon keputusan satu *level* (tingkat). [14]. Contoh penerapan *OneR* untuk mendeteksi penyakit *liver* dengan menggunakan cara membuat satu aturan untuk masing-masing atribut dalam *data training* kemudian melilih *rule* dengan *error* terkecil yang di sebuat aturan (*one rule*). *Pseudo-code* dari algoritma *OneR* ditunjukkan pada gambar 2.4 di bawah ini,

```
For each attribute (att),
    For each value of that att, make a rule as follows;
        Count how often each value of class appears
        Find the most frequent class
        Make the rule assign that class to this value of the att
    Calculate the total error of the rules of each att
Choose the att with the smallest total error.
```

**Gambar 2. 4 *Pseudo-code OneR***

### 2.2.10 *Random Forest*

*Random Forest* adalah salah satu metode yang digunakan untuk mengklasifikasikan dan registrasi, metode ini merupakan kesimpulan dari metode pembelajaran menggunakan pohon keputusan sebagai *base classifier* yang dibangun dan dikombinasikan. Dalam metode *Random Forest* ada tiga aspek penting yaitu melakukan *bootstrap* untuk membangun pohon prediksi, masing-masing pohon keputusan dibuat untuk *predictor* secara acak, dan *Random Forest* melakukan prediksi dengan mengkombinasikan hasil darisetiap pohon keputusan dengan cara *majority vote* untuk klasifikasi atau rata- rata untuk registrasi [15]. Contoh penerapan algoritma *Random Forest* untuk prediksi curah hujan dengan metode berbasis klasifikasi dan registrasi dimanaterdapat proses agregasi pohon keputusan, metode ini di pilih karena menghasilkan kesalahan yang lebih rendah dan memberikan akurasi yang bagus dan efektif untuk mengatasi data yang tidak lengkap. Rumus *Random Forest* ditunjukkan pada persamaan di bawah ini:

$$A(x) = \arg\max \{\sum_{i=1}^{z} Q(A(B_1\theta_k) = J\} \ldots.(7)$$

Yaitu:

A(x) adalah model hutan acak

J mewakili variabel katagori target

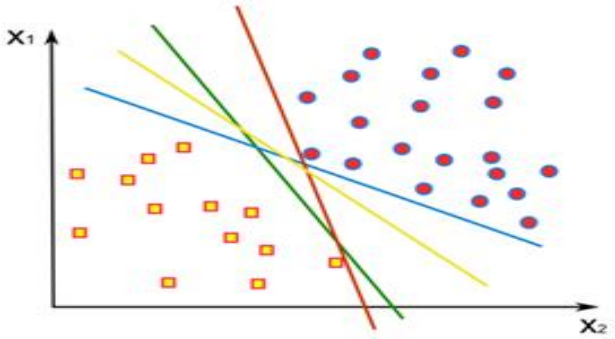
Q adalah fungsi karakteristik

Arg maxJ adalah nilai Ketika fungsi di dalam kurung kurawal memberikan nilai maksimun

### 2.2.11 *Support Vector Machine (SVM)*

SVM adalah mesin yang mengggunakan vektro sebagai pendukung atau penanda untuk membagi data menjadi dua kelompok. Dengan kata lain SVM adalah Teknik yang menggunakan dua titik (dua vektor),di mana kedua titik ini

akan membentuk garis pemisah (atau batas dalam tiga dimensi atau lebih).Garis batas atau yang terbentuk dari kedua vektor ini disebut *hyperplane*. Ilustrasi *hyperplane* ditunjukkan pada gambar 2.5 di bawah ini[16].




**Gambar 2. 5 *D Hyperplane***

Dapat dilihat bahwa ada dua kelompok data atau klasifikasi. Tugas SVM adalah membagi dua kelompok ini sebaik mungkin.Harus ada dua titik yang jadi patokan *hyperplane* tersebut,yang disebut *support vector* dan hasil SVM yang bagus bergantung pada vektor dukungan.

### 2.2.12 *Multi Layer Perceptron (MLP)*

MLP adalah algoritma popular dari keluarga jaringan saraf. Ini adalah algoritma yang kuat bekerja sebagai metode pelatihan yang diawasi,dan menggunakan sampel data dengan keluaran yang diketahui [17].

Proses dan karakteristik dari MLP adalah sebagai berikut:

1. *Feedforward Neural Network*: MLP termasuk dalam kategori jaringan saraf feedforward, di mana sinyal data mengalir maju dari lapisan input ke lapisan output tanpa ada feedback loop atau koneksi mundur.
2. Aktivasi Neuron: Setiap neuron di lapisan tersembunyi dan lapisan output menggunakan fungsi aktivasi yang non-linear untuk mengenalkan sifat non-linear dalam pemodelan data. Fungsi aktivasi seperti ReLU (Rectified Linear Unit), Sigmoid, atau Hyperbolic Tangent (tanh) sering digunakan.
3. Pelatihan dengan Backpropagation: MLP dilatih menggunakan algoritma backpropagation, yang melibatkan iterasi untuk mengoptimalkan bobot dan bias antar neuron dalam jaringan. Proses ini berdasarkan perhitungan gradien dari fungsi kesalahan (error) yang dihasilkan pada lapisan output,

kemudian dilakukan penyesuaian bobot dan bias secara mundur ke lapisan tersembunyi.

4. Universal Approximator: MLP secara teori dianggap sebagai universal approximator, artinya dengan jumlah lapisan tersembunyi dan neuron yang cukup, MLP dapat memetakan fungsi kompleks dari input ke output. Ini berarti MLP memiliki kapasitas untuk memodelkan data yang sangat kompleks dan memiliki tingkat generalisasi yang baik.

Keunggulan MLP:

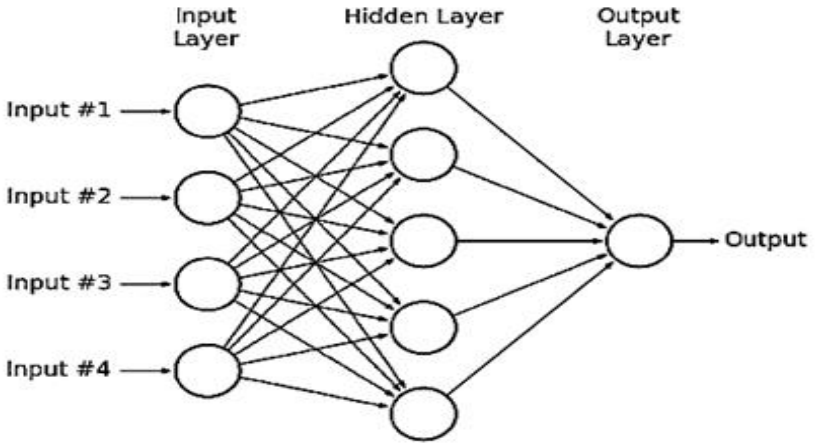
1. Kemampuan Representasi yang Kuat: MLP dapat memodelkan hubungan yang sangat kompleks antara input dan output, sehingga dapat digunakan untuk berbagai tugas seperti klasifikasi, regresi, dan pemrosesan data lainnya.
2. Kemampuan Belajar dari Data: MLP dapat belajar dari data pelatihan untuk menemukan pola-pola dan representasi yang lebih baik untuk memahami data secara keseluruhan.
3. Fleksibilitas: MLP dapat digunakan pada berbagai jenis data, termasuk data kontinu, kategorikal, dan gambar.

Kelemahan MLP:

1. Kompleksitas Pengaturan: Pengaturan arsitektur MLP seperti jumlah lapisan tersembunyi, jumlah neuron, dan fungsi aktivasi memerlukan pemilihan yang hati-hati dan eksperimen untuk mendapatkan hasil terbaik.
2. Pelatihan yang Memerlukan Waktu Lama: Pelatihan MLP dapat memerlukan waktu yang lama dan memakan sumber daya komputasi yang besar, terutama pada dataset besar dan arsitektur yang kompleks.
3. *Overfitting*: MLP rentan terhadap *overfitting* jika tidak diatur dengan baik atau jika terlalu kompleks untuk ukuran dataset yang tersedia.

MLP adalah salah satu algoritma *deep learning* yang paling kuat dan populer dalam *machine learning*. Dengan kemampuannya yang kuat untuk memodelkan data yang kompleks, MLP telah digunakan secara luas dalam berbagai aplikasi, termasuk pengenalan pola, pemrosesan bahasa alami, penglihatan komputer, dan masih banyak lagi.

MLP memiliki *neuron* yang terhubung ke node penghubung ,disusun dengan setidaknya tiga lapisan:lapisan masukan,lapisan tersembunyi,dan lapisan keluaran. Sementara itu algoritma NN yang lebih baru seperti CNN dan RNN muncul dalam hal akurasi,tetapi tidak tersedia dalam perangkat lunak WEKA standar.Gambar 2.6 menunjukkan bagaimana MLP bekerja dengan banyak lapisan.




**Gambar 2. 6 *Multi Layer Perceptron.***

### 2.2.13 Validasi Silang (Cross-Validation)

Validasi Silang (Cross-Validation) merupakan metode yang umum digunakan untuk mengukur kinerja model K-Nearest Neighbors (KNN) dengan membagi dataset menjadi subset pelatihan dan pengujian. Validasi silang membantu mengatasi masalah overfitting dan memberikan perkiraan yang lebih dapat diandalkan tentang kemampuan generalisasi model KNN.

Dalam validasi silang, dataset dibagi menjadi k-fold, di mana setiap iterasi, salah satu subset diambil sebagai data pengujian, sementara subset lainnya digunakan untuk pelatihan model. Ini berarti setiap sampel akan digunakan sebagai data pengujian dan juga sebagai bagian dari data pelatihan pada suatu titik.

Metode ini memberikan perkiraan yang lebih stabil dan objektif tentang kinerja model KNN. Dengan menggunakan variasi data pengujian, kita dapat mendapatkan perkiraan yang lebih akurat tentang sejauh mana model KNN dapat melakukan generalisasi pada data yang belum pernah dilihat sebelumnya. Validasi silang membantu mengurangi kemungkinan overfitting dan memastikan bahwa model tidak hanya melakukan memorisasi pada data pelatihan.

Hasil validasi silang KNN dapat digunakan untuk memilih parameter K

yang optimal dan juga untuk membandingkan kinerja model KNN dengan metode klasifikasi lainnya. Metode ini juga memberikan informasi yang berguna tentang variabilitas model terhadap dataset yang berbeda. Namun, perlu diingat bahwa validasi silang juga memerlukan waktu dan komputasi yang lebih tinggi karena melibatkan multiple pelatihan dan pengujian model.

Dengan menggunakan validasi silang, kita dapat lebih memahami sejauh mana model KNN dapat melakukan prediksi yang akurat dan dapat diandalkan pada data baru. Validasi silang membantu meningkatkan kepercayaan terhadap kinerja model KNN dan memastikan bahwa model tersebut memiliki kemampuan generalisasi yang baik.

### 2.2.14 *Confusion Matrix*

Untuk mengukur dan mengevaluasi performansi dari suatu model pada klasifikasi maka diperlukan adanya metode *confusion matrix*. Dalam pengukuran ini untuk mengetahui apakah *classifier* dapat mengelompokkan data kedalam kelas dengan baik. Hasil klasifikasi data yang dihasilkan dari suatu model memiliki hasil yang berbeda sesuai dengan tabel ketentuan pada *confusion matrix*. Berikut adalah tabel *confusion matrix* yang digunakan.

**Tabel 2. 2 *Confusion Matrix* [41]**

| Kelas Prediksi | Kelas Sebenarnya | |
|---|---|---|
| | *True* | *False* |
| *True* | *True Positive* (TP) | *False Positive* (FP) |
| *False* | *False Negative* (FN) | *True Negative* (TN) |

Tabel 2.2 menggambarkan hasil dari *confusion matrix* pada klasifikasi gambar. Keterangan dalam tabel tersebut dapat dijelaskan sebagai berikut.

a. *True Positive* (TP) : pada *true positive* hasil yang didapatkan adalah true pada kelas prediksi dan hasil klasifikasinya pada kelas sebenarnya juga true. Karena keduanya true maka disebut true positive.
b. *False Positive* (FP) : pada hasil ini akan menghasilkan kelas prediksi yang salah (false), namun hasil pengujian untuk kelas yang sebenarnya benar (atau positif).

c. *False Negative* (FN) : hasil perbandingan ini menunjukkan kelas prediksi yang masih salah dan negative menunjukkan kelas yang sebenarnya pada citra juga salah.
d. *True Negative* (TN) : pada kondisi ini kelas prediksi pada citra memiliki hasil yang benar, namun pada saat pengujian hasil kelas yang sebenarnya pada citra justru salah.

### 2.2.15 *Software WEKA*

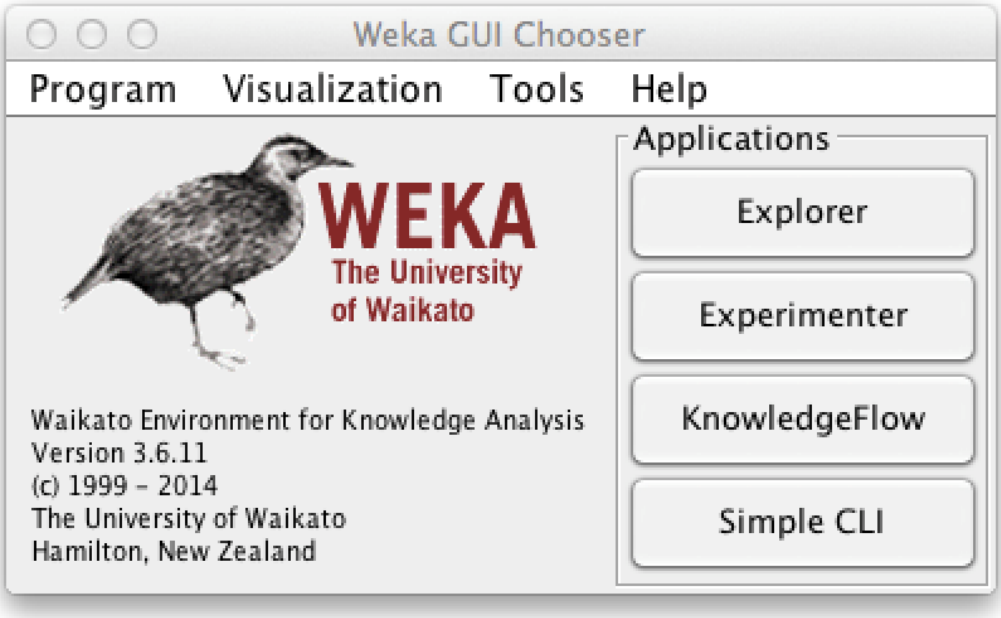
Weka adalah sebuah perangkat lunak sumber terbuka (open-source) yang populer dalam bidang analisis data dan pembelajaran mesin. Weka menyediakan beragam algoritma pembelajaran mesin, eksplorasi data, dan visualisasi data yang dapat digunakan untuk pemrosesan data, pemodelan prediktif, dan evaluasi model. Perangkat lunak ini ditulis dalam bahasa pemrograman Java dan menyediakan antarmuka pengguna grafis yang intuitif.

Salah satu keunggulan Weka adalah kemampuannya dalam menganalisis dan memproses berbagai jenis data, termasuk data terstruktur maupun tidak terstruktur. Weka mendukung berbagai format file, termasuk format ARFF (Attribute-Relation File Format) yang khusus digunakan dalam lingkungan Weka. Selain itu, Weka juga menyediakan alat visualisasi yang memungkinkan pengguna untuk memahami dan menganalisis data dengan lebih baik melalui grafik, diagram, dan visualisasi interaktif.

Weka juga menawarkan berbagai algoritma pembelajaran mesin yang dapat digunakan untuk melakukan tugas seperti klasifikasi, regresi, clustering, asosiasi, dan seleksi fitur. Algoritma-algoritma ini mencakup Decision Trees, Naive Bayes, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), dan banyak lagi. Pengguna dapat memilih algoritma yang sesuai dengan kebutuhan analisis data mereka dan mengkonfigurasi parameter yang relevan.

Selain itu, Weka memiliki fitur validasi silang (cross-validation) yang berguna untuk menguji dan membandingkan kinerja model secara objektif. Validasi silang memungkinkan pengguna untuk membagi dataset menjadi subset pelatihan dan pengujian, dan melakukan evaluasi kinerja model secara konsisten. Hal ini membantu dalam pemilihan parameter yang optimal dan memastikan bahwa model yang dikembangkan mampu melakukan generalisasi dengan baik.

Dengan fitur-fitur yang luas, antarmuka yang ramah pengguna, dan fleksibilitas dalam analisis data, Weka menjadi pilihan populer bagi para peneliti, analis data, dan praktisi dalam bidang pembelajaran mesin dan analisis data. Keberadaan komunitas Weka yang aktif juga memberikan dukungan dan pembaruan terkini dalam pengembangan software ini.




**Gambar 2. 7 Tampilan Software WEKA [19]**