BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Perkembangan Penyakit Daun pada Tanaman Padi

Padi merupakan salah satu komoditas tanaman pangan pokok yang berperan penting dalam ketahanan pangan nasional. Untuk menghasilkan beras yang siap dikonsumsi, padi harus melalui proses penggilingan. Beras sendiri menjadi komponen utama dalam pola makan masyarakat Indonesia. Meskipun jumlah penduduk terus meningkat setiap tahun, produksi padi justru mengalami penurunan. Penurunan produksi tersebut disebabkan oleh berbagai faktor, salah satunya adalah serangan penyakit pada tanaman padi (Lestari, 2019). Tindakan pencegahan terhadap penyakit pada tanaman padi sangat diperlukan guna meminimalkan kerugian yang dapat menyebabkan penurunan produktivitas. Upaya pencegahan tersebut dapat dilakukan melalui proses identifikasi berbagai jenis penyakit yang berpotensi menyerang tanaman padi. Beberapa jenis penyakit utama yang umum ditemukan pada tanaman padi antara lain blas daun (*Pyricularia oryzae*), tungro, hawar pelepah, kerdil rumput, hawar daun bakteri (*Xanthomonas oryzae pv. oryzae*), dan gosong palsu. Dengan mengenali karakteristik masing-masing penyakit tersebut, strategi pencegahan yang efektif dan berkelanjutan dapat diterapkan untuk menjaga kesehatan serta produktivitas tanaman padi (Walascha et al., 2022). Jenis penyakit tersebut dapat diidentifikasi melalui pengamatan visual atau analisis laboratorium dengan memperhatikan setiap tanda yang muncul pada permukaan daun. Tabel 2.1 menampilkan gejala-gejala atau tanda-tanda penyakit yang biasanya terlihat pada daun tanaman padi (Susanti et al., 2018).

Tabel 2.1 Jenis penyakit pada tanaman padi

No	Jenis Penyakit	Gejala
1	Blas Daun (*Pyricularia oryzae*)	Ditandai dengan munculnya bercak berbentuk belah ketupat dengan ujung runcing. Bagian tengah bercak berwarna putih kelabu, sedangkan tepinya tampak coklat. Pada serangan berat, bercak dapat bergabung dan menyebabkan daun mengering.

Tabel 2.2 Jenis penyakit pada tanaman padi

No	Jenis Penyakit	Gejala
2	Tungro	Menunjukkan perubahan warna daun dari hijau menjadi kekuningan hingga oranye. Pertumbuhan tanaman menjadi terhambat sehingga tanaman tampak kerdil.
3	Hawar Daun Bakteri	Gejala awal berupa bercak kecil berwarna coklat tua. Bercak kemudian meluas dengan tepi berwarna coklat dan bagian tengah berwarna kuning pucat hingga kelabu. Pada serangan berat, daun menjadi layu.
4	Kerdil Rumput	Ditandai dengan munculnya bercak kecil berwarna coklat tua pada daun muda. Seiring waktu, bercak membesar dengan bagian tengah berwarna putih kotor. Tanaman yang terinfeksi tampak lebih pendek dan pertumbuhannya tidak normal.
5	Hawar Pelepah	Muncul bercak tidak beraturan pada pelepah daun. Bagian tepi bercak berwarna kemerahan, sedangkan bagian tengah berwarna coklat muda. Pada tingkat infeksi lanjut, pelepah dapat membusuk.
6	Gosong Palsu (*Ustilaginoidea virens*)	Jamur penyebab penyakit ini berkembang di dalam sekam padi dan mengubah endosperma menjadi sklerotium berukuran besar.

Berdasarkan gejala yang muncul pada tanaman padi, setiap jenis penyakit memiliki karakteristik yang berbeda dan menimbulkan tanda-tanda khas pada permukaan daun, seperti perubahan warna menjadi coklat. Untuk meminimalkan kerugian yang dialami petani akibat serangan penyakit, diperlukan tindakan pencegahan yang efektif melalui proses identifikasi dan pengenalan jenis penyakit secara cepat serta efisien. Salah satu pendekatan yang dapat digunakan untuk mengenali gejala pada permukaan daun adalah dengan menerapkan teknik identifikasi dan klasifikasi penyakit tanaman padi berbasis komputer. Pendekatan ini berperan penting dalam proses deteksi dini terhadap penyakit, sehingga

tindakan pengendalian yang tepat dapat dilakukan guna menjaga kesehatan tanaman dan mencegah penyebaran lebih lanjut.

2.2 Keaslian dan Kontribusi Penelitian

Proses deteksi penyakit pada daun tanaman padi melibatkan beberapa tahapan utama, yaitu akuisisi data, praproses, ekstraksi fitur, segmentasi, dan klasifikasi penyakit. Dalam penelitian ini, fokus utama diberikan pada tahap ekstraksi fitur dengan memanfaatkan karakteristik warna, tekstur, tepi, dan bentuk (Rozikin et al., 2023). Hasil proses ekstraksi menghasilkan dataset fitur warna, tekstur, tepi, dan bentuk, yang selanjutnya digabungkan menjadi satu himpunan fitur gabungan. Namun, fitur-fitur hasil penggabungan tersebut memiliki beberapa kelemahan, antara lain adanya fitur yang tidak relevan, fitur yang bersifat redundan, waktu komputasi yang tinggi, serta potensi penurunan akurasi model klasifikasi. Permasalahan tersebut perlu diatasi melalui proses seleksi fitur pada dataset gabungan agar hanya fitur yang optimal yang digunakan dalam proses klasifikasi.

Kebaruan penelitian ini terletak pada perumusan kerangka komputasi yang secara komprehensif mengevaluasi kinerja tiga algoritma seleksi fitur bio-inspiratif algoritma tersebut yaitu SMO, PSO, dan ACO pada domain klasifikasi citra daun padi yang bersifat multi-kelas dan memiliki karakteristik visual yang kompleks. Berbeda dari studi sebelumnya yang umumnya menguji algoritma tersebut secara terpisah, penelitian ini mengintegrasikan ketiganya ke dalam satu pipeline eksperimen untuk memperoleh pemahaman yang lebih mendalam mengenai perilaku optimasi fitur pada data citra pertanian. *Novelty* berikutnya muncul dari penggabungan seleksi fitur dengan skema *hyperparameter tuning* terstruktur pada dua model klasifikasi utama, yakni *Support Vector Machine (SVM)* dan *Random Forest (RF),* yang menghasilkan peningkatan performa signifikan melalui pemilihan konfigurasi model yang lebih presisi. Selain itu, penelitian ini menekankan evaluasi multidimensional yang tidak hanya berfokus pada akurasi prediksi, tetapi juga menganalisis efisiensi komputasi dari setiap kombinasi metode. Pendekatan yang holistik ini memberikan kontribusi metodologis penting bagi pengembangan sistem diagnosis penyakit tanaman berbasis machine learning yang layak diterapkan pada konteks dunia nyata,

khususnya pada platform pertanian presisi dengan keterbatasan sumber daya komputasi.

Perbandingan antara hasil penelitian ini dengan beberapa penelitian sebelumnya disajikan pada Tabel 2.2. Tabel tersebut memuat perbedaan dan kesamaan dalam hal metode, data yang digunakan, serta hasil yang diperoleh, sehingga dapat memberikan gambaran posisi dan kontribusi penelitian ini terhadap penelitian-penelitian terdahulu.

Tabel 2.3 Perbedaan penelitian yang dilakukan sebelumnya

No	Author	Metode	Deskripsi
1	(Petchiammal et al., 2023)	CNN dengan *transfer learning* menggunakan arsitektur VGG16, MobileNet, Xception, dan ResNet34	Penelitian ini membandingkan empat arsitektur CNN, yaitu VGG16, MobileNet, Xception, dan ResNet34, dalam klasifikasi penyakit daun padi. Hasil menunjukkan bahwa arsitektur ResNet34 memberikan akurasi tertinggi sebesar **97,50%**.
2	(Lamba et al., 2023)	CNN, LSTM, dan GAN	Penulis mengusulkan model **GCL**, yang merupakan kombinasi dari GAN, CNN, dan LSTM. GAN digunakan untuk augmentasi data, CNN berperan dalam ekstraksi fitur penyakit, dan LSTM digunakan sebagai model klasifikasi. Model ini menghasilkan akurasi rata-rata sebesar **97%**.
3	(Sakhamuri & Kumar, 2022)	DCNN–CS (*Deep Convolutional Neural Network* dan *Cuckoo Search*)	Penelitian ini mengusulkan algoritma DCNN untuk klasifikasi penyakit daun padi dengan optimasi parameter menggunakan algoritma *Cuckoo Search* (CS). Hasil eksperimen menunjukkan akurasi tertinggi mencapai **99%**.

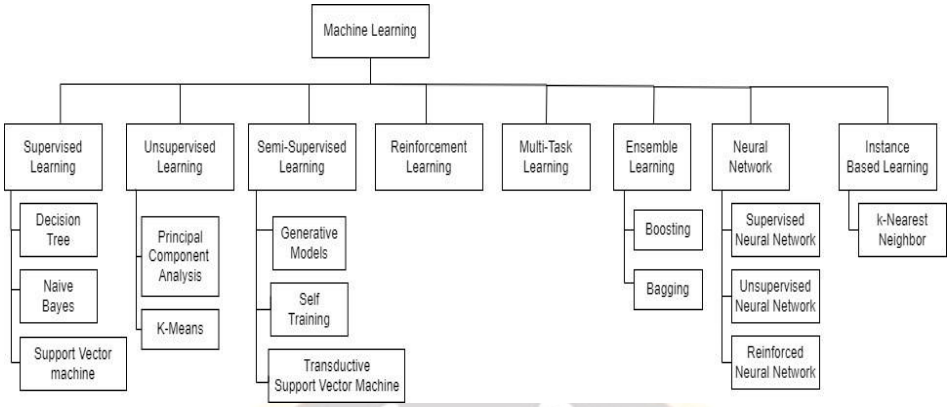
Tabel 2.4 Perbedaan penelitian yang dilakukan sebelumnya

No	Author	Metode	Deskripsi
4	(Neware, 2022)	KNN dan SVM	Penelitian ini membandingkan algoritma K-Nearest Neighbor (KNN) dan Support Vector Machine (SVM) dalam klasifikasi penyakit daun padi. Hasil menunjukkan rata-

			rata akurasi sebesar **95%** untuk KNN dan **98%** untuk SVM.
5	Proposed metode	SMO, PSO, ANT, RF, dan SVM plus hyperparameter tunning	Penelitian yang diusulkan menerapkan dan membandingkan tiga metode seleksi fitur bio-inspiratif (SMO, PSO, dan ACO). Selain itu, dilakukan integrasi *hyperparameter tuning* pada SVM dan RF. Evaluasi performa mencakup aspek akurasi serta efisiensi komputasi untuk mendukung penerapan di dunia nyata.

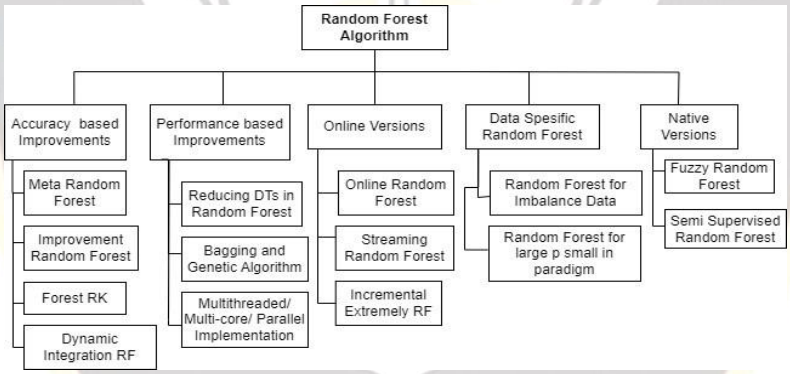
2.3 *Machine Learning*

Saat ini, teknologi *machine learning* telah banyak diterapkan di berbagai bidang. Contohnya, *machine learning* digunakan untuk pengenalan suara, pengendalian kendaraan otonom, prediksi penyakit pada tanaman, dan berbagai aplikasi lainnya (Vanneschi & Silva, 2023). *Machine learning* merupakan bidang studi yang mempelajari bagaimana komputer dapat mensimulasikan proses belajar manusia serta mengembangkan kemampuan untuk memperoleh pengetahuan dan keterampilan baru. Selain itu, *machine learning* juga berfokus pada metode yang memungkinkan komputer mengidentifikasi pengetahuan yang telah ada dan secara berkelanjutan meningkatkan kinerja serta kemampuan dalam melaksanakan tugas-tugas tertentu (Hua & Learning, 2010). Performa *machine learning* bergantung pada algoritma dan karakteristik data yang digunakan, karena setiap algoritma memiliki keunggulan dan keterbatasan yang perlu disesuaikan dengan jenis data yang diolah. Para ilmuwan data menegaskan bahwa tidak ada satu algoritma pun yang paling ideal untuk menyelesaikan semua jenis permasalahan. Pemilihan algoritma sangat bergantung pada karakteristik permasalahan yang ingin diselesaikan, jumlah serta jenis variabel yang digunakan, dan model yang paling sesuai untuk tujuan analisis. Gambar 2.1 merupakan beberapa algoritma yang umum digunakan dalam *machine learning* (Batta, 2018).



Gambar 2.1 Algoritma *machine learning* (Mahesh, 2018)

Random Forest merupakan salah satu teknik dalam *ensemble learning* yang berkembang pesat dalam beberapa tahun terakhir. Algoritma ini banyak diterapkan dalam bidang *data mining*, yang secara umum terbagi menjadi dua klasifikasi utama, yaitu deskriptif dan prediktif. Perkembangan dan penerapan *Random Forest* dapat digambarkan melalui ilustrasi pada Gambar 2.2 (Kulkarni, 2013).



Gambar 2.2 Perkembangan algoritma *Random Forest* (Y Kulkarni dan K Sinha, 2013)

Algoritma *machine learning* bekerja dengan prinsip dasar mempelajari pola dari data, yang biasanya disebut sebagai *data pelatihan* (*training data*). Setelah itu, algoritma diuji menggunakan *data uji* (*testing data*) untuk mengevaluasi performanya. Berdasarkan tujuan dan karakteristik data yang digunakan, algoritma *machine learning* dapat dibedakan menjadi beberapa jenis, antara lain:

2.3.1 *Supervised Learning*

Jenis algoritma ini menghasilkan fungsi yang memetakan *input* ke *output* yang telah diketahui sebelumnya. Salah satu formulasi umum dari *supervised learning* adalah masalah klasifikasi, dalam hal ini algoritma belajar memetakan *input* ke salah satu dari beberapa kelas yang telah ditentukan. Sebagai contoh, seorang pelajar dapat mempelajari perilaku suatu fungsi dengan mengamati pasangan *input–output* yang telah tersedia.

2.3.2 *Unsupervised Learning*

Pada jenis ini, algoritma bekerja dengan sekumpulan data *input* tanpa label atau kategori sebelumnya. Tujuannya adalah mengelompokkan data ke dalam klaster yang memiliki kemiripan tertentu berdasarkan pola yang ditemukan dalam data.

2.3.3 *Semi-Supervised Learning*

Metode ini menggabungkan data berlabel dengan data tanpa label. Tujuannya adalah membangun fungsi atau pengklasifikasi yang mampu memanfaatkan kedua jenis data tersebut untuk meningkatkan akurasi model.

2.3.4 *Reinforcement Learning*

Algoritma *Reinforcement Learning* mempelajari cara mengambil keputusan berdasarkan interaksi dengan lingkungan. Setiap tindakan yang dilakukan menghasilkan umpan balik (*feedback*) berupa penghargaan atau hukuman, yang digunakan untuk menyempurnakan strategi pengambilan keputusan di masa berikutnya.

Pemahaman terhadap jenis-jenis algoritma *machine learning* ini sangat penting untuk menentukan pendekatan yang paling tepat sesuai dengan tipe data dan tujuan pembelajaran yang diinginkan (Nasteski, 2017).

2.4 Pengumpulan dan pra pemrosesan data

Komputer mampu mengidentifikasi serta mengklasifikasikan beragam jenis penyakit daun pada tanaman padi melalui beberapa tahapan standar. Secara umum, tahapan yang dilaksanakan guna mengidentifikasi dan mengklasifikasikan jenis penyakit daun pada tanaman padi adalah sebagai berikut:

1. Pengumpulan Data: Mengumpulkan data yang diperlukan untuk analisis penyakit daun pada tanaman padi.
2. Pra Pemrosesan (Praproses): Membersihkan, merapihkan, dan menormalkan data mentah agar siap untuk tahap berikutnya.
3. Ekstraksi Fitur: Mengambil ciri-ciri penting dari data yang dapat mewakili karakteristik penyakit daun.
4. Seleksi Fitur: Memilih ciri-ciri yang paling relevan dan signifikan untuk proses klasifikasi.
5. Pembangunan Model Klasifikasi: Membangun model komputer yang dapat mengklasifikasikan penyakit daun berdasarkan ciri-ciri yang diambil.
6. Evaluasi Model: Menguji dan mengevaluasi kinerja model klasifikasi terhadap data yang belum pernah dilihat sebelumnya.

Dengan mengikuti langkah-langkah tersebut, komputer dapat berhasil mengidentifikasi serta mengklasifikasikan jenis-jenis penyakit daun pada tanaman padi dengan akurasi dan efisiensi yang lebih baik (Nisar et al., 2020).

Pengumpulan data pada umumnya dilakukan dengan memperoleh dataset dari berbagai penyedia data daring, seperti *PlantVillage* (www.plantvillage.com), *GitHub* (www.github.com), dan *Kaggle* (www.kaggle.com) (Ridhovan et al., 2022) Selain itu, data juga dapat dikumpulkan dengan memotret secara langsung daun padi yang terinfeksi penyakit (Patil, 2021). Secara umum, pengumpulan data dapat dilakukan melalui dua metode. Metode pertama dilakukan dengan mengkondisikan lingkungan pengambilan citra. Daun ditempatkan pada area yang telah dikondisikan guna memperoleh citra objek daun sesuai kebutuhan (R. Sharma et al., 2020) (Zhao et al., 2020). Metode kedua dilakukan dengan mengambil citra dalam kondisi lingkungan alami, seperti di area persawahan milik petani. Pengambilan citra di lingkungan alami ini bertujuan untuk memastikan bahwa model yang dikembangkan dapat beroperasi secara optimal dalam kondisi nyata dengan tingkat akurasi yang tinggi (Appalanaidu & Kumaravelan, 2021).

Dataset yang telah dikumpulkan perlu melalui tahap prapemrosesan data agar siap digunakan dalam proses pembentukan model. Berdasarkan berbagai

literatur, terdapat sejumlah metode prapemrosesan yang disesuaikan dengan karakteristik dataset. Tahap ini dapat mencakup beberapa langkah, seperti pemangkasan (*cropping*) dan penyesuaian ukuran citra dari ukuran asli menjadi ukuran tertentu sesuai kebutuhan analisis data (Parven et al., 2020).

Prapemrosesan dataset dapat dilakukan melalui berbagai metode, antara lain transformasi ruang warna dari format RGB ke format HSV (Ramesh & Vydeki, 2019), transformasi ruang warna dari RGB menjadi *grayscale* (Payal & Kukana, 2020), pemisahan kanal warna RGB menjadi kanal R, kanal G, dan kanal B (Jayanthi & Shashikumar, 2019), penambahan jumlah citra dalam dataset melalui teknik *augmentasi* (Hasan et al., 2019), serta penghapusan *noise* dan peningkatan kualitas citra menggunakan teknik *median filter* (Gayathri Devi & Neelamegam, 2019). Selain itu, berbagai teknik prapemrosesan lain juga diterapkan oleh sejumlah peneliti untuk meningkatkan akurasi hasil analisis citra.

2.5 Ekstraksi Fitur

Hasil dari prapemrosesan berupa dataset citra yang telah memiliki ukuran seragam dan terbebas dari gangguan *noise*. Selanjutnya, dataset citra tersebut menjalani tahap ekstraksi fitur. Proses ekstraksi fitur dilakukan untuk memperoleh nilai-nilai yang merepresentasikan karakteristik spesifik citra, seperti warna, tekstur, tepi, dan bentuk. Secara umum, ekstraksi fitur mencakup beberapa aspek utama, yaitu ekstraksi fitur warna, tekstur, bentuk, dan ciri tepi (Azim et al., 2021). Teknik ekstraksi fitur merupakan cara untuk mendapatkan nilai dari dataset citra (Vishnoi et al., 2020). Dalam konteks ekstraksi fitur warna, pendekatan yang dilakukan mencakup pengambilan nilai rata-rata warna, nilai standar deviasi warna, dan nilai *skewness* warna (Azim et al., 2021). Untuk menghitung nilai rata-rata warna, nilai standar deviasi warna, kurtosis, dan nilai skewness, dapat menggunakan persamaan (Sachar & Kumar, 2021).

$$m = \frac{1}{M\ X\ N} \sum_{X=1}^{M} \sum_{y=1}^{n} M_{xy} \tag{2.1}$$

$$SD = \sqrt{\frac{1}{M\ X\ N} \sum_{x=1}^{M} \sum_{y=1}^{N} (M_{xy} - m)^2} \tag{2.2}$$

$$KT = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} (M_{xy} - m)^2}{(X\ x\ Y)\ x\ SD^4} 3 \tag{2.3}$$

$$SK = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} (M_{xy} - m)^3}{(M \; x \; N) \; x \; SD^3} \quad (2.4)$$

Secara umum, ekstraksi fitur bentuk dilakukan untuk memperoleh nilai-nilai seperti soliditas, eksentrisitas, diameter, luas, pusat massa, panjang sumbu minor, dan panjang sumbu mayor (Vishnoi et al., 2020) (Thyagharajan & Kiruba Raji, 2019). Nilai-nilai fitur bentuk tersebut, seperti area, *aspect ratio*, orientasi, keliling (*perimeter*), serta panjang sumbu mayor dan minor, dapat dihitung menggunakan persamaan yang dikemukakan (Sharif et al., 2018).

$$Area = \sum_{i=1}^{n} \sum_{j=1}^{m} A[i,j] \quad (2.5)$$

$$Aspect\ ratio = \frac{width}{height} \quad (2.6)$$

$$Orientation = tan^{-1} \left(\frac{y}{x}\right) \quad (2.7)$$

$$Major\ axis\ and\ minor\ axis = x_1 + x_2, \sqrt{(x_1 + x_2)^1 - d} \quad (2.8)$$

2.6 Perkembangan Algoritma Seleksi Fitur

Seleksi fitur merupakan proses pemilihan atribut atau variabel yang relevan dalam analisis data. Secara umum, terdapat dua pendekatan utama dalam teknik seleksi fitur. Pendekatan pertama didasarkan pada penilaian peringkat fitur, yaitu setiap fitur dievaluasi berdasarkan tingkat akurasi relatifnya. Pendekatan kedua melibatkan penggunaan algoritma seleksi fitur yang memanfaatkan metode reduksi dimensi (Mahapatra & Sahu, 2022). Berbagai penelitian telah dilakukan untuk menerapkan seleksi fitur sebelum tahap pemodelan menggunakan algoritma klasifikasi. Salah satu penelitian menerapkan algoritma *Ant Colony Optimization (ACO)* untuk memilih fitur hasil ekstraksi ciri bentuk, morfologi, tekstur, dan warna. Proses klasifikasi selanjutnya dilakukan menggunakan algoritma *Support Vector Machine (SVM)***,** yang menghasilkan akurasi rata-rata sebesar *87,4%* (Ali Jan Ghasab et al , 2015). Penelitian lainnya menggunakan algoritma *Particle Swarm Optimization (PSO)* untuk memilih fitur hasil ekstraksi tekstur, bentuk, dan warna. Proses klasifikasi dilakukan dengan algoritma *Fuzzy-Relevance Vector Machine (FRVM)***,** yang mencapai tingkat akurasi sebesar 99,5% (VijayaLakshmi & Mohan, 2016).

Pemilihan fitur menggunakan algoritma *Crow Search Optimization Algorithm (CSA)* pada hasil ekstraksi fitur warna kemudian dilanjutkan dengan proses klasifikasi menggunakan algoritma *Fast Fuzzy C-Means (FFCM)***.** Hasil penelitian menunjukkan bahwa metode ini memberikan akurasi yang lebih baik (Anter et al., 2019). Hasil ekstraksi fitur *vein***,** geometri, dan tekstur digabungkan menjadi satu fitur gabungan. Fitur gabungan tersebut kemudian diseleksi menggunakan algoritma *Artificial Bee Colony Optimization (ABC)***.** Selanjutnya, hasil seleksi fitur diklasifikasikan menggunakan algoritma *Multi-Kernel with Parallel Deep Learning (MK-PDL)***,** dengan rata-rata akurasi sebesar 86,57% (Kumar et al., 2019). kstraksi fitur warna dan tekstur menghasilkan fitur *fusion*, yang kemudian diseleksi menggunakan algoritma *Particle Swarm Optimization (PSO)***.** Proses klasifikasi menghasilkan akurasi rata-rata sebesar 98% (Singh, 2019).

Hasil ekstraksi fitur *Subtractive Pixel Adjacency Matrix* (SPAM) menghasilkan 686 fitur. Untuk mereduksi jumlah fitur tersebut, digunakan algoritma *Enhanced Spider Monkey Optimization* (ESMO), sehingga diperoleh 82 fitur. Fitur-fitur tersebut kemudian diklasifikasikan menggunakan beberapa algoritma, yaitu *Support Vector Machine* (SVM), *K-Nearest Neighbor* (KNN), *Linear Discriminant Analysis* (LDA), dan *ZeroR*. Hasil terbaik diperoleh pada kombinasi algoritma ESMO dan SVM dengan akurasi sebesar 92,12% (Andrushia & Patricia, 2020). Hasil ekstraksi fitur warna, bentuk, dan tekstur digabungkan menjadi fitur gabungan. Fitur gabungan tersebut kemudian diseleksi menggunakan algoritma *Artificial Bee Colony Optimization (ABC),* dan hasil seleksi diklasifikasikan menggunakan algoritma SVM, dengan akurasi rata-rata sebesar 93,01% (Andrushia & Patricia, 2020). Fitur bentuk, tekstur, dan warna diekstraksi untuk menghasilkan nilai-nilai fitur bentuk, tekstur, dan warna. Nilai-nilai tersebut kemudian diseleksi menggunakan algoritma genetika, dan hasilnya diklasifikasikan menggunakan algoritma SVM dengan akurasi rata-rata sebesar 90,40% (Kaur et al., 2020).

2.7 Algoritma Spyder Monkey Optimization (SMO)

Nama *swarm* digunakan untuk merujuk pada sekelompok makhluk, seperti semut, ikan, burung, rayap, dan lebah madu, yang menunjukkan perilaku kolektif.

Swarm Intelligence merupakan pendekatan metaheuristik dalam bidang teknik yang terinspirasi dari fenomena alam serta digunakan untuk menyelesaikan permasalahan optimasi. Algoritma optimasi berbasis *swarm* meliputi *Particle Swarm Optimization (PSO)*, *Ant Colony Optimization (ACO)*, *Artificial Bee Colony Optimization (ABC)*, *Spider Monkey Optimization (SMO)*, dan lainnya.

Spider Monkey Optimization (SMO) merupakan algoritma *swarm intelligence* yang terinspirasi oleh perilaku monyet dalam mencari makanan. SMO bekerja melalui proses iteratif dan kolaboratif yang didasarkan pada prinsip *trial and error*. Algoritma SMO terdiri atas tujuh tahapan, yaitu *Initialization of the Population*, *Local Leader Phase*, *Global Leader Phase*, *Local Leader Learning*, *Global Leader Learning*, *Local Leader Decision*, dan *Global Leader Decision* (Bansal et al., 2013).

2.7.1 Initialization Of The Population

Tahap inisialisasi pada algoritma *Spider Monkey Optimization* (SMO) menghasilkan populasi awal *spider monkey* sebanyak N, yang terdistribusi secara seragam dengan individu *spider monkey* dilambangkan sebagai SM_idengan $i = 1,2, \dots, N$, yang merupakan vektor berdimensi D. Nilai Dmenunjukkan jumlah variabel dalam permasalahan optimisasi, dan setiap SM_imerepresentasikan *spider monkey* ke-i dalam populasi. Setiap *spider monkey* sesuai dengan satu solusi potensial dalam konteks permasalahan yang sedang diselesaikan. Inisialisasi setiap SM_idilakukan menggunakan persamaan berikut:

$$SM_{ij} = SM_{minj} + U(0,1)\ x\ (SM_{maxj} - SM_{minj}) \quad (2.9)$$

Dimana $SM_{maxj}\ dan\ SM_{minj}$ adalah batas dari SM_i ke arah j^{th}dan $U(0,1)$ adalah bilangan acak yang terdistribusi secara merata dengan range $[0,1]$.

2.7.2 Local Leader

Dalam tahap *Local Leader* pada algoritma *Spider Monkey Optimization* (SMO), setiap *Spider Monkey* (SM) memperbarui posisinya dengan mempertimbangkan informasi yang diperoleh dari pengalaman *Local Leader* serta pengalaman anggota kelompok lokal. Nilai kecocokan (*fitness*) dari posisi baru yang dihasilkan kemudian dihitung. Apabila nilai kecocokan posisi baru lebih tinggi dibandingkan dengan nilai kecocokan posisi sebelumnya, maka SMtersebut akan memperbarui posisinya dengan posisi baru tersebut.

Persamaan yang digunakan untuk memperbarui posisi *Spider Monkey* ke-i, yang merupakan anggota kelompok lokal ke-k, dinyatakan sebagai berikut:

$$SMnew_{ij} = SM_{ij} + U(0,1)\ x\ (LL_{kj} - SM_{ij}) + U(-1,1)\ x\ (SM_{rj} - SM_{ij}) \quad (2.10)$$

Di mana SM_{ij} merupakan dimensi ke-j dari SM ke-i, LL_{kj} merepresentasikan dimensi ke-j dari posisi pemimpin kelompok lokal ke-k. SM_{rj} adalah dimensi j^{th} dari $SM\ r^{th}$ yang dipilih secara acak dalam kelompok k^{th} sehingga $r \neq 1$, $U(0,1)$ adalah bilangan acak yang terdistribusi merata antara 0 dan 1.

2.7.3 Global Leader

Setelah menyelesaikan tahap pemimpin lokal (*local leader*), langkah selanjutnya adalah tahap pemimpin global *(global leader).* Dalam tahap *global leader*, semua *Spider Monkey SM* memperbarui posisi mereka berdasarkan informasi dari *global leader* serta pengalaman anggota grup lokal. Persamaan untuk memperbarui posisi pada tahap ini dapat dirumuskan sebagai berikut:

$$SMnew_{ij} = SM_{ij} + U(0,1)\ x\ (GL_{kj} - SM_{ij}) + U(-1,1)\ x\ (SM_{rj} - SM_{ij}) \quad (2.11)$$

Dimana GL_j merepresentasikan dimensi ke-j dari posisi pemimpin global dan $j \in \{1, 2, ..., D\}$ adalah indeks yang dipilih secara acak.

Dalam tahapan ini, posisi *spider monkeys* (SM_i) diperbarui berdasarkan peluang probabilitas yang dihitung menggunakan nilai kecocokan (*fitness*) mereka. Melalui pendekatan ini, kandidat yang memiliki nilai kecocokan yang lebih baik akan memiliki peluang yang lebih tinggi untuk meningkatkan performanya. Probabilitas dapat dihitung dengan menggunakan ekspresi berikut (meskipun ada variasi lain, tetapi umumnya berfungsi sebagai fungsi dari nilai kecocokan):

$$prob_i = 0.9\ x\ \frac{fitness_i}{max\_fitness} + 0{,}1 \quad (2.12)$$

Dalam hal ini, *fitness* merupakan nilai kecocokan dari SM ke-i dan *max_ fitness* adalah nilai kecocokan maksimal dalam kelompok. Setelahnya, nilai kecocokan dari posisi baru yang dihasilkan dihitung dan dibandingkan dengan nilai kecocokan dari posisi sebelumnya. Kemudian, SM akan mengadopsi posisi yang lebih baik berdasarkan hasil perbandingan tersebut. Selanjutnya, kesesuaian

posisi SM yang baru dihasilkan dihitung dan dibandingkan dengan yang lama dan mengadopsi posisi yang lebih baik.

2.7.4 Global Leader Learning

Dalam tahapan ini, posisi pembelajaran pemimpin global *(global leader learning)* diperbarui melalui penerapan seleksi *greedy* di dalam populasi. Ini berarti bahwa posisi *Spider Monkey* yang memiliki kebugaran terbaik dalam populasi dipilih sebagai posisi pemimpin global yang baru. Setelah itu, dilakukan pemeriksaan untuk menentukan apakah posisi pemimpin global telah diperbarui atau tidak. Jika tidak, maka nilai *Global Limit Count* akan ditambah satu.

2.7.4.1 Local Leader Learning

Dalam tahapan ini, posisi pemimpin lokal *(local leader)* diperbarui dengan menerapkan seleksi *greedy* di dalam kelompok tersebut. Artinya, posisi *Spider Monkey* yang memiliki nilai kecocokan terbaik dalam populasi dipilih sebagai posisi pemimpin lokal yang baru. Selanjutnya, posisi pemimpin lokal yang baru diperbandingkan dengan yang sebelumnya, dan jika posisi pemimpin lokal tidak diperbarui, maka nilai *Local Limit Count* akan bertambah satu.

2.7.4.2 Local Leader Decision

Apabila terdapat keputusan pemimpin lokal *(local leader decision)* yang tidak diperbarui hingga mencapai batas yang sebelumnya ditetapkan, yang dikenal sebagai Batas Pemimpin Lokal *(Local Leader Limit*), maka seluruh anggota kelompok tersebut akan melakukan pembaruan terhadap posisi mereka. Pembaruan ini dapat dilakukan melalui inisialisasi acak atau dengan memanfaatkan informasi yang digabungkan dari pemimpin global dan pemimpin lokal melalui penggunaan Persamaan yang relevan.

$$SMnew_{ij} = SM_{ij} + U(0,1)\ x\ (GL_j - SM_{ij}) + U(0,1)\ x\ (SM_{ij} - LL_{kj}) \quad (2.13)$$

Dari persamaan diatas, terlihat bahwa dimensi pembaruan pada SM ini lebih cenderung kepada pemimpin global *(global leader)* dari pada pemimpin lokal *(local leader)*.

2.7.4.3 Global Leader Decision

Dalam tahapan ini, posisi *global leader decision* dipantau dan jika tidak diperbarui hingga sejumlah iterasi yang telah ditentukan yang disebut batas

pemimpin global *(Global Leader Limit),* maka pemimpin global akan membagi populasi menjadi kelompok-kelompok kecil. Awalnya, populasi dibagi menjadi dua kelompok, lalu tiga kelompok, dan seterusnya hingga mencapai jumlah maksimum kelompok. Proses pemimpin lokal *(Local Leader)* dimulai untuk memilih pemimpin lokal dalam setiap kelompok yang baru terbentuk. Apabila jumlah kelompok maksimum terbentuk dan bahkan posisi pemimpin global tidak mengalami pembaruan, maka pemimpin global akan menggabungkan semua kelompok menjadi satu kelompok tunggal. Dengan cara ini, algoritma yang diusulkan mengambil inspirasi dari struktur fungsi-fungsi dalam perilaku *Spider Monkey (SM).*

Dalam dataset fitur penyakit daun padi, terdapat banyak fitur yang dapat menurunkan performa model dan meningkatkan beban komputasi, sehingga diperlukan seleksi fitur. Seleksif fitur menggunakan SMO dibutuhkan dataset fitur untuk mencari dan memilih fitur optimal. Hasil dari algoritma seleksif itu berupa dataset yang terdapat fitur optimal. Dataset ini akan digunakan sebagai input pada model klasifikasi *Random Forest* dan *Support Vector Machine*. Hasil prediksi menggunakan algoritma SVM akurasinya rata-rata 90.40% (Kaur, et al., 2020).s

2.8 Algoritma *Particle Swarm Optimization (PSO)*

Particle Swarm Optimization (PSO) merupakan salah satu algoritma optimasi berbasis populasi yang dirancang untuk menyelesaikan persoalan fungsi-fungsi kompleks yang tidak dapat diselesaikan secara deterministik. Diperkenalkan pertama kali oleh Kennedy dan Eberhart pada tahun 1995, algoritma ini terinspirasi oleh perilaku sosial organisme yang hidup dalam kawanan, seperti burung dan ikan, yang secara kolektif mencari sumber makanan melalui interaksi sederhana antar individu. Pendekatan ini menempatkan PSO dalam keluarga *swarm intelligence*, yaitu agen-agen individu (partikel) secara adaptif memanfaatkan informasi lokal dan global untuk mencari solusi optimal dalam ruang pencarian (Sengupta et al., 2019).

Populasi partikel dalam PSO merepresentasikan himpunan solusi potensial. Setiap partikel memiliki posisi dan kecepatan, yang secara berulang diperbarui berdasarkan pengalaman terbaik yang pernah dicapainya (*personal best* atau *pbest*) serta pengalaman terbaik seluruh partikel dalam populasi (*global best*

atau *gbest*). Perubahan posisi dipengaruhi oleh komponen kognitif (pengetahuan individu), sosial (pengaruh lingkungan), dan inersia (momentum pergerakan sebelumnya). Tujuan dari komponen ini adalah untuk menyeimbangkan dua aspek penting dalam pencarian solusi: eksplorasi ruang solusi yang luas dan eksploitasi solusi terbaik yang telah ditemukan (Jain et al., 2018). Persamaan matematis dasar dari pembaruan kecepatan dan posisi partikel dalam PSO dinyatakan sebagai berikut:

$$v_{ij}^{k+1} = w.v_{ij}^{k} + c_p.r_p.(pbest_{ij} - x_{ij}^{k} +) + c_g.r_g.(gbest_i - x_{ij}^{k}) x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \quad (2.14)$$

Dimana x_{ij}^{k}: Posisi partikel ke-j pada dimensi ke-i di iterasi ke-k, v_{ij}^{k}: kecepatan partikel, w: bobot inersia yang mengontrol kesetablian pergerakan, c_p dan c_g koefesien akselerasi kognitif sosial, r_p dan r_g: bilangan acak yang didistribuskan secara uniform.

Beberapa penelitian menunjukkan bahwa kinerja PSO sangat dipengaruhi oleh pengaturan parameter seperti bobot inersia, batas kecepatan, dan koefisien akselerasi. Pengaturan yang tepat dapat meningkatkan stabilitas algoritma dan mencegah konvergensi prematur, yaitu kondisi di mana partikel terlalu cepat terjebak pada solusi lokal yang suboptimal. Untuk mengatasi hal tersebut, berbagai varian PSO telah dikembangkan, termasuk adaptasi bobot inersia secara dinamis, pembentukan sub-populasi, strategi pemetaan parameter, hingga integrasi teknik pemutakhiran non-linier. Secara historis, PSO telah dikembangkan ke dalam beberapa versi, seperti Standard PSO (SPSO), *Constricted* PSO, dan *Multi-objective* PSO (MOPSO), serta telah di-*hybrid* dengan algoritma lain seperti *Genetic Algorithm (GA),* Differential Evolution (DE), dan Simulated Annealing (SA). Tujuan dari hibridisasi ini adalah meningkatkan kapasitas eksplorasi dan memperluas cakupan penerapan PSO pada permasalahan yang lebih kompleks seperti optimasi multiobjektif, diskrit, dan berbatasan (Houssein et al., 2021).

Pada penerapannya, PSO terbukti efektif untuk berbagai jenis permasalahan seperti klasifikasi, seleksi fitur, perencanaan rute, peramalan, kontrol sistem, pengenalan pola, dan pengolahan citra. Keunggulan utamanya terletak pada kemudahan implementasi, fleksibilitas adaptasi, serta efisiensi dalam

menemukan solusi mendekati optimal tanpa membutuhkan informasi turunan dari fungsi objektif. Seiring dengan meningkatnya kebutuhan terhadap sistem komputasi cerdas yang mampu menangani data berukuran besar dan kompleks, PSO tetap menjadi salah satu algoritma optimasi yang relevan dan kompetitif dalam berbagai domain, baik di bidang teknik, biomedis, maupun kecerdasan buatan.

2.9 Algoritma *Ant Colony Optimization (ACO)*

Ant Colony Optimization (ACO) merupakan salah satu algoritma optimasi *metaheuristik* yang dikembangkan dengan meniru perilaku sosial semut dalam mencari jalur tercepat menuju sumber makanan. Algoritma ini pertama kali diperkenalkan oleh Marco Dorigo pada tahun 1992 melalui studinya tentang pemecahan masalah *Traveling Salesman Problem (TSP).* Sejak saat itu, ACO berkembang menjadi pendekatan yang banyak digunakan untuk menyelesaikan berbagai masalah optimasi kombinatorial, seperti penjadwalan, routing, dan pemodelan jaringan. Prinsip dasar ACO terinspirasi dari cara semut membentuk jalur optimal melalui mekanisme komunikasi kimiawi yang disebut feromon. Saat semut menjelajah dari sarang ke sumber makanan, mereka meninggalkan jejak feromon di sepanjang lintasan yang mereka lewati. Semakin banyak semut melewati jalur tersebut, konsentrasi feromon akan semakin tinggi, dan ini akan menarik semut lain untuk mengikuti jalur yang sama. Akibatnya, secara kolektif, semut akan cenderung menemukan jalur terpendek secara efisien tanpa adanya arahan langsung (Akhand et al., 2020).

Dalam perspektif komputasi evolusioner, *Ant Colony Optimization* (ACO) adalah algoritma optimasi meta-heuristik yang meniru mekanisme kolektif koloni semut nyata untuk menemukan rute paling efisien. Pada level operasional, setiap agen buatan (semut) membangun solusi secara inkremental dengan memanfaatkan skema probabilistik yang dipandu oleh dua komponen utama, yaitu intensitas feromon sebagai representasi pengalaman historis dan visibilitas heuristik yang umumnya diformulasikan sebagai kebalikan jarak atau informasi domain yang sepadan. Proses eksplorasi dan eksploitasi dalam ACO diatur melalui dua mekanisme pembaruan feromon: local pheromone updating yang diterapkan selama agen membentuk solusi untuk mempertahankan keberagaman jalur dan

mencegah stagnasi prematur, serta global pheromone updating yang diberikan pada jalur terbaik setelah seluruh agen menyelesaikan satu siklus konstruksi, sehingga memperkuat arah pencarian menuju solusi optimal. Seiring perkembangan riset, ACO berevolusi menjadi beberapa varian algoritmik, di antaranya *Ant System (AS)* sebagai model dasar yang memperkenalkan pondasi perilaku kolektif semut, *Ant Colony System (ACS)* yang mengoptimalkan keseimbangan antara eksploitasi solusi terbaik dan percepatan konvergensi melalui aturan pembaruan feromon diferensial, serta *MAX–MIN Ant System (MMAS)* yang menerapkan batasan maksimum dan minimum feromon untuk mengontrol dinamika pencarian dan mengurangi potensi konvergensi prematur pada ruang solusi yang kompleks.

Selain masalah diskrit seperti TSP dan penjadwalan, ACO juga berhasil diterapkan pada domain optimasi berkelanjutan, pemodelan sistem kompleks, klasifikasi, dan pemrosesan data besar, bahkan hingga pengembangan robotik dan jaringan komunikasi. Dalam penelitian terkini, ACO juga telah di-*hybrid* dengan algoritma lain seperti PSO, GA, dan algoritma *deep learning* untuk meningkatkan efektivitas dan fleksibilitasnya terhadap tantangan optimasi multiobjektif dan masalah dimensi tinggi.

2.10 Algoritma *Support Vector Machine (SVM)*

SVM adalah algoritma pembelajaran terawasi yang bekerja dengan mencari *hyperplane* optimal untuk memisahkan kelas dalam ruang fitur berdimensi tinggi. *Support Vector Machine (SVM)* merupakan salah satu algoritma *supervised learning* yang diperkenalkan oleh Cortes dan Vapnik (1995), dirancang untuk memisahkan data ke dalam kelas yang berbeda dengan mencari sebuah *hyperplane* optimal di ruang fitur (CORINNA CORTES, 1995). *Hyperplane* optimal ini dipilih sedemikian rupa sehingga memiliki *margin* maksimum terhadap titik-titik data terdekat dari masing-masing kelas, yang dikenal sebagai *support vectors*. *Margin* yang lebih besar umumnya memberikan generalisasi model yang lebih baik terhadap data baru (Vapnik & N., 1995). Dalam kasus data yang terpisahkan secara linear (*linearly separable*), SVM bertujuan memaksimalkan margin di antara dua kelas. Secara matematis, permasalahan optimisasi SVM dapat dirumuskan sebagai:

$\min_{w,b} \frac{1}{2}\|w\|^2$ dengan kendala $y_i(w.x_i + b) \geq 1, \quad \forall_i$ (2.15)

Dalam model *Support Vector Machine* (SVM), ω merepresentasikan vektor bobot yang berfungsi untuk menentukan arah dan kemiringan bidang pemisah (*hyperplane*). Sementara itu, b menunjukkan nilai bias yang berperan dalam menggeser posisi bidang pemisah agar memperoleh margin optimal. Setiap x_i merupakan vektor fitur ke – i yang merepresentasikan data masukan dalam ruang fitur, sedangkan $y_i \epsilon\{-1, +1\}$ menunjukkan label kelas dari data tersebut, yang menandakan dua kategori yang akan dipisahkan oleh model SVM.

Pada banyak kasus, data tidak dapat dipisahkan secara linear di ruang aslinya. Untuk mengatasi hal ini, SVM memanfaatkan kernel trick untuk memproyeksikan data ke ruang fitur berdimensi lebih tinggi di mana data menjadi lebih mudah dipisahkan (Schölkopf & Smola, 2002). Fungsi kernel yang umum digunakan antara lain:

1. Linier Kernel: $K(x, x') = x.x'$
2. Polynomial Kernel: $K(x, x') = (\gamma x.x' + r)^d$
3. Radial Basis Function (RBF): $K(x, x') = exp(-\gamma||x - x'||^2)$
4. Sigmoid Kernel: $K(x, x') = \tanh(\gamma x.x' + r)$

Support Vector Machine (SVM), pendekatan soft margin dikembangkan untuk menangani kondisi data yang tidak sepenuhnya dapat dipisahkan secara linear. Konsep ini memperkenankan terjadinya pelanggaran margin melalui variabel slack ε_i, sehingga masalah optimasinya dirumuskan dengan meminimalkan fungsi objektif $(1/2)\ \|w\|^2 + C\sum\varepsilon_i$ di bawah kendala $y_i\ (w \cdot x_i + b) \geq 1 - \varepsilon_i$ dan $\varepsilon_i \geq 0$. Secara geometris, SVM berupaya mencari hyperplane pemisah dengan margin maksimum, yang dihitung sebagai $2/\|w\|$, sehingga nilai $\|w\|$ menentukan seberapa luas margin yang dihasilkan. Pada soft margin, beberapa titik data diperbolehkan memasuki area margin atau bahkan salah klasifikasi melalui nilai ε_i yang menunjukkan tingkat pelanggaran dari setiap titik data. Parameter C mengatur besarnya penalti terhadap pelanggaran tersebut: nilai C besar mendorong model membatasi kesalahan secara ketat sehingga margin menjadi sempit dan berpotensi menimbulkan *overfitting*, sementara C yang lebih kecil memperluas margin tetapi memungkinkan lebih banyak pelanggaran sehingga berisiko menghasilkan underfitting. Selain itu, performa SVM sangat dipengaruhi oleh pengaturan

hyperparameter γ (gamma) pada kernel non-linear seperti RBF, polynomial, dan sigmoid; gamma besar membuat *decision boundary* menjadi sangat kompleks, sedangkan gamma kecil menghasilkan batas keputusan yang lebih halus. Pemilihan jenis kernel menjadi aspek fundamental karena menentukan bentuk transformasi ruang fitur yang digunakan, sehingga memiliki dampak langsung terhadap kapasitas model dalam menangkap pola non-linear pada ruang data.

Keunggulan utama SVM terletak pada kemampuannya menggunakan fungsi kernel untuk memproyeksikan data ke ruang berdimensi lebih tinggi, sehingga data yang tidak terpisahkan secara linear dapat dipisahkan secara efektif. *Hyperparameter* utama pada SVM meliputi parameter regularisasi **C**, parameter kernel seperti **γ** (gamma) untuk *Radial Basis Function (RBF)* kernel, dan jenis kernel yang digunakan. Nilai C mengontrol tingkat penalti terhadap kesalahan klasifikasi, sedangkan γ mengatur pengaruh satu data latih terhadap pembentukan *decision boundary* (Fowler, 2000). Penelitian terkait membandingkan metode tuning hyperparameter SVM seperti *grid search*, *random search*, dan *Bayesian optimization*, dan menemukan bahwa pendekatan Bayesian lebih efisien (Saradhi, 2025).

2.11 Random Forest Classification

Random Forest (RF) adalah *classifier* yang terdiri dari kumpulan *classifier* berstruktur pohon $\{h(x, \Theta_k), k = 1, \dots\}$ dimana $\{\Theta_k\}$ adalah vektor acak yang terdistribusi secara identik dan independen dan setiap pohon memberikan *unit vote* untuk kelas paling populer pada input x. deifinis ini menunjukan *Random Forest* merupakan kombinasi dari *classifer* berstruktur pohon. setiap pohon ditanam berdasarkan kumpulan sampel pelatihan dan variabel acak, variabel acak yang sesuai dengan pohon ke-k dilambangkan sebagai Θ_k antara dua variabel acak ini independen dan terdistribusi secara identik, menghasilkan pengklasifikasi $h(x, \Theta_k)$ dimana x adalah input vektor. Setelah k kali berjalan, kita mendapatkan urutan pengklasifikasi $\{h_1(x), h_2(x), \dots h_k(x)\}$, dan gunakan ini untuk membentuk lebih dari satu sistem model klasifikasi, hasil akhir dari sistem ini hilang oleh suara mayoritas biasa, fungsi keputusannya adalah sebagai berikut (Liu et al., 2012).

$$H(x) = arg \underset{y}{max} \sum_{i=1}^{k} I(h_i(x) = Y) \quad (2.16)$$

Dimana $H(x)$ adalah kombinasi model klasifikasi, h adalah model pohon keputusan tunggal, Y adalah variabel keluaran, $I(.)$ adalah fungsi indikator. Untuk variabel input yang diberikan, setiap pohon memiliki hak untuk memilih hasil klasifikasi terbaik.

Dalam RF, fungsi margin digunakan untuk mengukur sejauh mana rata-rata jumlah suara di X, Y untuk kelas yang tepat melebihi kelas yang salah, tentukan fungsi margin sebagai berikut:

$$mg(X,Y) = av_k I(h_k(X) = Y) - max_{j \neq Y}\, av_k\, I(h_k(X) = j) \quad (2.17)$$

Semakin besar nilai margin, semakin tinggi akurasi prediksi klasifikasi, dan semakin konfiden dalam klasifikasi. Penentuan kesalahan generalisasi dari classifier dirumuskan dengan persamaan berikut.

$$PE^* = P_{X,Y}(mg(X,Y) < 0) \quad (2.18)$$

Bila jumlah pohon keputusan cukup besar $h_x(X) = h(X, \Theta_k)$ mematuhi Hukum Kuat Bilangan Besar. Leo Breiman telah membuktikan dua kesimpulan. Salah satunya adalah RF tidak *over-pas* tapi benar-benar menghasilkan nilai yang membatasi kesalahan generalisasi. Alasannya adalah dengan jumlah pohon keputusan yang bertambah, hampir pasti untuk semua urutan $\Theta_1; \ldots PE^*$ konvergen.

$$P_{X,Y}(P_\theta(h_k(X,\theta) = Y) - max_{j \neq Y} P_\theta(h(X,\theta) = j) < 0) \quad (2.19)$$

Selain itu batas atas adanya kesalahan generalisasi

$$PE^* \leq \bar{\rho}(1 - s^2)/s^2 \quad (2.20)$$

s adalah kekuatan himpunan pengklasifikasi $\{h(X, \theta)\}$, $\bar{\rho}$ adalah nilai rata-rata korelasi. Ini menunjukkan bahwa kesalahan generalisasi RF bergantung pada dua aspek: satu adalah kekuatan individu pohon di hutan, yang lain adalah korelasi antara pohon-pohon ini.

2.12 Hyperparameter Tuning

Hyperparameter tuning merupakan proses pencarian kombinasi parameter terbaik yang tidak dipelajari secara langsung dari data, tetapi ditetapkan sebelum proses pelatihan model (Bischl et al., 2023). Pemilihan *hyperparameter* yang tepat berpengaruh signifikan terhadap kinerja model, baik dari sisi akurasi prediksi maupun kemampuan generalisasi. Strategi tuning yang umum digunakan antara lain *grid search*, *random search*, dan *Bayesian optimization* (Snoek et al., 2012).

Dalam konteks klasifikasi, penggunaan teknik validasi silang (*cross-validation*) sering dipadukan untuk mengevaluasi performa setiap kombinasi parameter (Wainer & Cawley, 2017).

Random Forest adalah algoritma *ensemble learning* yang memanfaatkan kombinasi banyak *decision tree* untuk meningkatkan akurasi dan mengurangi risiko *overfitting* (Liu et al., 2012). Model ini memiliki beberapa *hyperparameter* penting, di antaranya jumlah pohon (*n_estimators*), jumlah fitur yang dipertimbangkan pada setiap percabangan (*max_features*), kedalaman maksimum pohon (*max_depth*), dan jumlah data minimum pada setiap *leaf node* (*min_samples_leaf*) (Louppe, 2015). *Tuning hyperparameter* RF dapat meningkatkan performa model secara signifikan, terutama pada dataset dengan kompleksitas tinggi. Beberapa metode optimasi yang digunakan untuk tuning RF meliputi *grid search*, *random search*, serta pendekatan berbasis *model-based optimization* (Probst et al., 2019).

2.13 Confussion Matrix

Confusion matrix adalah salah satu metode pengukuran keputusan yang paling banyak digunakan. *Confusion matrix* adalah representasi dari nilai aktual dan nilai prediksi suatu data. (Soleh et al., 2021)

Tabel 2.5 Confusion matrix

Tabel *confusion matrix*		
Nilai sebenarnya	**Nilai Sebenarnya**	
	Kelas 0 *(Crack)*	**Kelas 1 *(Scratch)***
Kelas 0 *(Crack)*	TP	FP
Kelas 1 *(Scratch)*	FN	TN

Pada Tabel 2.3 tiap kolom pada matriks adalah contoh kelas prediksi, sedangkan tiap baris mewakili kejadian dikelas yang sebenarnya. Tiap kolom dan baris pada confusion matrix, masing-masing memiliki 0 dan 1, 0 berarti negatif/false dan suatu berarti positif/true. Pada confusion matrix, ada empat bagian dari hasil klasifikasi. Empat bagian itu sebagai berikut :

TP = *True positif* yaitu model memprediksi penyakit A dan kenyataanya penyakit A

TN = *True negatif* yaitu model memprediksi penyakit A, kenyataannya penyakit B

FP = *False positif* yaitu model memprediksi bukan penyakit A, kenyataanya bukan penyakit A

FN = *False negatif* yaitu model memprediksi penyakit A, kenyataanya bukan penyakit A

Model yang telah dibangun selanjutnya dievaluasi untuk mengetahui kinerja dari model tersebut. Secara umum ada dua teknik yang digunakan dalam mengukur kinerja model tersebut yaitu *confusion matrix* dengan parameter yang dievaluasi yaitu akurasi, presisi, *recall* dan *F-1 score*. Untuk mendapatkan nilai hasil evaluasi tersebut dapat menggunakan rumus berikut (Meiriyama et al., 2022).

a. Akurasi (*Accuracy*)

Akurasi memberitahukan seberapa akuratnya sistem dalam pengklasifikasian. *Accuracy* adalah resiko prediksi yang benar (positif dan negatif) terdapat pada keseluruhan data atau tingkat kedekatan antara nilai yang dipredisikan dengan nilai sebenarnya

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.21)$$

b. Presisi (*Precision*)

Presisi mengukur sejauh mana prediksi positif yang dibuat oleh model adalah benar. Presisi memberikan informasi tentang tingkat kesalahan positif palsu (prediksi positif yang salah).

$$Presisi = \frac{TP}{TP+FP} \quad (2.22)$$

c. Recall

Recall memberitahukan keberhasilan sistem ketika informasi ditemukan kembali. Recall adalah rasio perbandingan antara prediksi benar (positif) dengan benar (positif) dan salah (negatif).

$$Recall = \frac{TP}{TP+FN} \quad (2.23)$$

d. F1-Score

F1-Score merupakan harmonic mean antara presisi dan *recall*. Metrik ini memberikan ukuran komprehensif tentang kinerja model dalam memprediksi secara akurat di kedua kelas positif dan negatif.

$$F-1\,Score = 2\,x\,\frac{Presisi\ x\ Recall}{Presisi+Recall} \quad (2.24)$$