BAB 2
LANDASAN TEORI

Dalam penelitian ini, digunakan beberapa literatur sebagai bahan acuan dalam mengembangkan penelitian skripsi ini. Dari telaah literatur yang ada, diambil beberapa teori-teori serta rumus matematika yang digunakan dalam mengembangkan penelitian ini.

2.1 Penelitian Terdahulu

Tabel 2.1 merupakan penelitian terdahulu yang menjadi dasar penelitian ini.

Tabel 2.1. Ringkasan Penelitian Terdahulu

Nama Peneliti & Tahun	**Judul Penelitian**	**Metode/Algoritma**	**Hasil Penelitian**
Belhaouari et al. (2023)	*A Bird's Eye View Feature Selection for High Dimensional Data*	*Bird Eye View* (BEV) *Feature Selection*	BEV meraih *accuracy* hingga 100% pada beberapa dataset dan mengurangi jumlah fitur secara signifikan.
Kurnia et al. (2023)	Seleksi Fitur dengan *Particle Swarm Optimization* pada Klasifikasi Penyakit *Parkinson* Menggunakan XGBoost	PSO + XGBoost + SMOTE	AUC meningkat dari 0.9250 (tanpa PSO) menjadi 0.9325 (dengan PSO).

Tabel 2.1. Ringkasan Penelitian Terdahulu (lanjutan)

Nama Peneliti & Tahun	**Judul Penelitian**	**Metode/Algoritma**	**Hasil Penelitian**
Ghatasheh et al. (2023)	*Modified Genetic Algorithm for Feature Selection and Hyperparameter Optimization: Case of XGBoost in Spam Prediction*	*Modified Genetic Algorithm* + XGBoost	*Accuracy* sebesar 92.67%, *geometric mean* sebesar 82.32%, dan reduksi fitur hingga kurang dari 10%.
Ghosh et al. (2021)	*A Comparative Study of Enhanced Machine Learning Algorithms for Brain Tumor Detection and Classification*	SVM, KNN, RF, XGBoost, Regresi Logistik, NB, SGD, *Gradient Boosting*, dan DT.	XGBoost unggul dengan *accuracy* sebesar 90% untuk klasifikasi multi-kelas tumor otak.

Penelitian oleh Belhaouari et al. (2023) menciptakan metode seleksi fitur baru untuk mengatasi maraknya data dengan fitur berdimensi tinggi. Sering kali data dengan fitur berdimensi tinggi memiliki fitur yang tidak relevan, *noise* atau *outlier* yang dapat menurunkan kinerja model dan memakan sumber daya komputasi. Pada penelitian tersebut diungkapkan bahwa metode *existing* seperti *ranking-based, swarm intelligence*, atau *nature-inspired meta-heuristics* memiliki limitasi dalam menangani fitur berdimensi tinggi. Oleh karena itu, diusunglah metode yang memanfaatkan *supervised evolutionary algorithm* yang dibungkus dalam konfigirasi *wrappper*[10].

Bird Eye View (BEV) terinspirasi dari cara burung mencari makanan di hutan dengan melihat dari ketinggian (perspektif luas). BEV menggunakan *Evolutionary Algorithms, Dynamic Markov Chain*, dan *Reinforcement Learning* untuk menyeimbangkan eksplorasi dan eksploitasi ruang pencarian fitur. Selain itu BEV juga memiliki *hyperparameter* yang lebih mudah untuk diatur. Tabel 2.2 dan Tabel 2.3 merupakan tabel komparasi antar algoritma dari algoritma-algoritma terdahulu.

Tabel 2.2. Hasil komparasi rata rata *accuracy* antar algoritma

Dataset	FSBACOM	TFSACO	IRRFSACO	TSHFS-ACO	PSO	ECLPSO	CSO	VLPSO	ERM-FS	**Proposed BEV**
Lung Cancer	82.81±1.54	88.84±2.24	86.67±3.13	88.62±0.84	78.77±1.53	77.91±1.98	87.72±2.93	87.60±1.20	93.25±0.01	**100.0±0.00**
11 Tumor	80.21±1.76	81.71±1.17	78.85±1.60	85.12±1.30	71.81±1.75	71.09±1.20	79.52±2.35	82.38±1.94	80.26±0.02	**87.00±3.49**
Leukemia 2	92.22±2.12	92.44±2.04	87.72±2.44	95.00±1.43	89.83±1.00	89.82±1.20	91.71±3.16	91.56±2.05	98.12±0.01	**100.0±0.00**
Prostate	83.98±1.90	88.73±2.02	91.05±2.31	91.57±1.21	86.00±1.49	85.46±1.41	88.99±2.68	88.48±1.93	96.12±0.01	**100.0±0.00**
Brain Tumor 2	67.21±4.48	72.33±3.29	72.58±4.58	76.08±3.68	61.99±2.91	63.20±2.60	80.44±6.28	70.29±5.25	89.23±0.03	**98.00±5.37**
Brain Tumor 1	74.83±1.78	74.04±3.33	63.88±4.53	71.42±3.97	73.73±2.21	73.87±2.37	79.93±3.09	70.58±2.78	83.26±0.04	**89.00±0.00**
9 Tumor	40.83±5.49	45.83±5.33	40.00±3.80	50.67±5.53	42.72±1.42	41.33±1.48	59.50±3.72	47.33±4.23	64.44±0.07	**64.80±7.04**
DLBCL	89.48±2.00	91.22±6.43	91.53±3.66	93.95±1.68	83.67±1.52	82.44±2.01	94.30±4.05	91.03±3.85	98.09±0.02	**100.0±0.00**
Leukemia 1	84.68±4.23	93.90±1.13	80.32±2.78	94.81±2.35	80.60±2.55	80.88±2.28	88.45±3.90	96.05±2.62	97.93±0.01	**100.0±0.00**
SRBCT	88.96±3.00	98.71±0.89	89.13±2.35	99.42±0.53	89.51±1.56	88.10±1.57	93.29±5.52	98.88±0.70	100.0±0.00	**100.0±0.00**

Tabel 2.3. Jumlah fitur terpilih rata-rata dalam 100 kali uji seleksi fitur

Dataset	PSO	ECLPSO	CSO	FSBACOM	TFSACO	IRRFSACO	TSHFS-ACO	VLPSO	ERM-FS	**Proposed BEV**
Lung Cancer	6234.7	5739.7	226.4	379.1	61.9	96.0	96.3	181.0	61.00	**12.1**
11 Tumor	6205.0	5731.7	588.6	339.6	93.7	146.0	145.9	204.4	292.60	**430.6**
Leukemia 2	5535.7	5115.6	88.6	363.0	32.2	56.0	55.9	42.7	15.30	**5.6**
Prostate	5193.7	4818.5	357.2	420.9	12.2	65.0	64.9	38.08	11.00	**6.4**
Brain Tumor 2	5117.2	4718.7	90.4	218.5	46.8	74.0	74.1	113.6	26.80	**6.5**
Brain Tumor 1	2917.2	2710.0	207.6	120.5	38.7	71.0	70.9	103.8	32.44	**7.3**
9 Tumor	2811.9	2605.5	220.3	123.9	79.9	89.0	89.2	87.6	89.40	**108**
DLBCL	2681.0	2491.3	30.1	173.9	17.4	45.0	45.0	31.9	14.20	**6.0**
Leukemia 1	2615.5	2427.9	170.1	130.1	37.8	45.0	44.5	7.0	22.80	**6.1**
SRBCT	1119.4	1054.8	85.4	35.8	49.8	43.0	42.5	23.4	31.00	**10.4**

Tabel 2.2 menunjukkan hasil komparasi algoritma terdahulu dalam 100 kali uji seleksi fitur (*mean* ± std). Berdasarkan Tabel 2.2 BEV secara konsisten mengalahkan algoritma terdahulu dalam segi rata-rata *accuracy* dan Tabel 2.3 menunjukkan bahwa BEV mampu menghasilkan jumlah rata-rata *subset* fitur yang lebih kecil. Metode ini berhasil mengungguli metode *existing* seperti PSO dan algoritma berbasis ACO.

Penelitian yang dilakukan oleh Kurnia et al. (2023) meningkatkan AUC klasifikasi penyakit Parkinson dengan menggunakan kombinasi teknik *feature selection*, penyeimbangan data, dan optimasi *hyperparameter*. Data yang digunakan diambil dari *UCI Machine Learning Repository* yang berisi 756 sampel dengan 753 fitur hasil ekstraksi sinyal suara pasien. Algoritma *Particle Swarm Optimization* (PSO) dipakai untuk mereduksi fitur dan masalah *imbalance class* diatasi dengan menerapkan teknik *Synthetic Minority Over-sampling Technique* (SMOTE). Model klasifikasi yang digunakan adalah *Extreme Gradient Boosting* (XGBoost). Evaluasi dilakukan menggunakan metrik *Area Under Curve* (AUC).

Tabel 2.4. Perbandingan Nilai AUC

Seleksi Fitur	**XGBoost**	**XGBoost + SMOTE**	**XGBoost + SMOTE + *Random Search***
Tanpa Seleksi Fitur	0.9250	0.9304	0.9366
Dengan Seleksi Fitur	0.9325	0.9412	0.9483

Hasil penelitian menunjukkan bahwa PSO berhasil mengurangi fitur sebesar 50.86% (dari 753 menjadi 370 fitur) dengan nilai *fitness* terbaik 0.0443. Tanpa seleksi fitur, model XGBoost menghasilkan AUC sebesar 0.9250. Dengan PSO, AUC meningkat menjadi 0.9325. Ketika SMOTE dan *hyperparameter tuning* diterapkan, AUC meningkat lebih signifikan dari 0.9366 (tanpa PSO) menjadi 0.9483 (dengan PSO) seperti yang ditampilkan pada Tabel 2.4. Kombinasi PSO, SMOTE, dan *hyperparameter tuning* terbukti mampu meningkatkan performa klasifikasi penyakit Parkinson[11].

Penelitian yang dilakukan oleh Ghatasheh et al.(2023) mengusulkan algoritma genetika yang dimodifikasi untuk melakukan seleksi fitur dan optimasi *hyperparameter* pada model XGBoost dalam pendeteksi spam pada media sosial. Gabungan metode ini berhasil mengurangi jumlah fitur hingga kurang dari 10% dari total fitur awal namun tetap meningkatkan performa klasifikasi. Validasi dilakukan menggunakan 50 kali *stratified 10-fold cross-validation*.

Tabel 2.5. Komparasi performa model berdasarkan berbagai metrik evaluasi

Metric	F10-P400-C240-G50	XGBoost + Chi^2	MNNB + Chi^2	KNN3 + Chi^2	LR + Chi^2	AdaBoost + Chi^2	DT + Chi^2
Accuracy	**92.67** (0.010)	92.06 (0.009)	88.77 (0.009)	87.17 (0.009)	91.34 (0.009)	92.46 (0.010)	91.38 (0.011)
GMean	**82.32** (0.030)	76.12 (0.032)	58.13 (0.044)	50.88 (0.049)	71.46 (0.037)	86.02 (0.030)	82.20 (0.028)
TPR	69.68 (0.050)	58.71 (0.049)	33.99 (0.051)	26.23 (0.056)	51.63 (0.046)	66.55 (0.049)	**70.77** (0.047)
PPV	84.54 (0.039)	91.58 (0.035)	**99.49** (0.012)	96.31 (0.051)	95.50 (0.028)	86.02 (0.041)	76.81 (0.042)
AUC	92.62 (0.018)	**93.79** (0.016)	**95.61** (0.012)	79.13 (0.029)	95.34 (0.012)	92.82 (0.018)	83.31 (0.024)
F1s	**76.28** (0.037)	71.42 (0.039)	50.46 (0.057)	40.73 (0.063)	66.71 (0.037)	74.9 (0.038)	73.56 (0.035)

Tabel 2.6. Performa model BERT

class	PPV (precision)	TPR (recall)	f1	support
'ham'	0.91	0.97	0.94	1058
'spam'	0.79	0.52	0.63	216
accuracy			**0.90**	1274
macro avg	0.85	0.75	0.78	1274
weighted avg	0.89	0.90	0.89	1274

Hasil pada Tabel 2.5 menunjukkan bahwa pendekatan ini menghasilkan *accuracy* sebesar 92.67% dan *geometric mean* sebesar 82.32%. Pendekatan ini juga mengungguli metode seleksi fitur tradisional seperti *Chi-Square* dan PCA, serta model pembelajaran mesin lainnya, termasuk model BERT berbasis *deep learning* pada Tabel 2.6[12].

Peneliti Ghosh et al.(2021) berhasil melakukan komparasi berbagai algoritma *Support Vector Machine* (SVM), *Logistic Regression*, *K-Nearest Neighbor* (KNN), *Naïve Bayes* (NB), *Decision Tree* (DT), *Random Forest* (RF), XGBoost, *Stochastic Gradient Descent* (SGD), dan *Gradient Boosting*. Data yang digunakan dalam penelitian ini terdiri dari 2 *dataset* yaitu *dataset* A berisikan *binary classification*, *dataset* A terdiri dari 982 gambar hasil pindai MRI yang terkena tumor dan 493 yang tidak terkena tumor, lalu *dataset* B berisikan *multiclass classification* terdiri dari 826 gambar hasil pindai MRI otak yang terkena *glioma tumor*, 822 gambar otak yang terkena *meningioma tumor*, 827 gambar otak yang terkena *pituitary tumor*, dan 395 gambar otak yang tidak terkena tumor.

Tabel 2.7. Komparasi algoritma *machine learning* pada *dataset* A

Model	**Accuracy**	**Recall**	**Precision**	**F1-score**
SVM	0.851	0.754	0.798	0.775
Regresi Logistik	0.846	0.730	0.800	0.763
KNN	0.845	0.804	0.756	0.779
Naïve Bayes	0.769	0.627	0.675	0.650
Decision Tree	0.865	0.825	0.788	0.806
Random Forest	0.908	0.952	0.811	0.876
SGD	0.840	0.709	0.807	0.755
XGBoost	0.894	0.888	0.818	0.852
Gradient Boosting	**0.924**	**0.944**	**0.850**	**0.895**

Tabel 2.8. Komparasi algoritma *machine learning* pada *dataset* B

Metric	**SVM**	**KNN**	**Random Forest**	**XGBoost**
Accuracy	0.85	0.77	0.89	0.90
Recall (Weighted Avg)	0.85	0.77	0.89	0.90
Precision (Weighted Avg)	0.87	0.82	0.89	0.90
F1-score (Weighted Avg)	0.85	0.78	0.89	0.90

Tabel 2.7 merupakan tabel perbandingan kinerja antar algoritma untuk klasifikasi biner dan Tabel 2.8 merupakan tabel perbandingan kinerja antar algoritma untuk klasifikasi *multiclass* Hasil penelitian yang ditunjukkan pada Tabel 2.7 dan Tabel 2.8 menunjukkan model XGBoost unggul dalam segi *accuracy*, *precision*, *recall*, dan *F1-score*.

2.2 Tumor Otak

Tumor otak merupakan kondisi dimana terdapat pertumbuhan sel abnormal di jaringan otak atau sekitarnya, yang dapat bersifat jinak (non-kanker) maupun ganas (kanker) [13]. Di antara berbagai jenis tumor otak, *meningioma*, *glioma*, dan *pituitary tumor* termasuk yang paling umum ditemukan. *Meningioma* berasal dari selaput pelindung otak (meningen), umumnya bersifat jinak namun dapat menekan struktur otak vital jika tidak terdeteksi dini [14]. *Glioma* muncul dari sel glial, sering kali bersifat agresif dan sulit diobati yang memiliki prognosis buruk [15]. Selain itu, *pituitary tumor* berkembang di kelenjar pituitari dan sering kali bersifat jinak namun dapat mengganggu produksi hormon. *pituitary tumor* dapat menyebabkan gangguan metabolik dan neurologis [16].

2.3 Augmentasi Data

Augmentasi data telah menjadi teknik penting dalam analisis gambar medis, khususnya untuk dataset terbatas. Penelitian oleh [17] mengidentifikasi bahwa augmentasi data dapat meningkatkan *accuracy* model hingga 15% pada dataset kecil ($<$1000 sampel).

Tabel 2.9. Perbandingan Rasio Augmentasi terhadap Accuracy pada Klasifikasi Citra Medis

Rasio Augmentasi	***Accuracy* / AUC**	**Sumber**
1x (Tanpa Augmentasi)	0.818–0.880	[18, 19]
2x	0.819–0.910	[18, 20]
3x	0.812–0.930	[19, 20]
4x	**0.980**	[20]
5x+	Menurun / Jenuh	[18, 19]

Berdasarkan penelitian dari Xue et al. (2021), peningkatan data sintetis menggunakan GAN hingga 50% dari jumlah data asli (*augmentation ratio* 1.5x) memberikan peningkatan *accuracy* terbaik pada klasifikasi histopatologi, sedangkan rasio yang lebih tinggi menyebabkan penurunan performa model [18]. Hal serupa ditemukan oleh Bhatia et al. (2024) dalam studi klasifikasi X-ray dada, di mana *accuracy* model mencapai titik optimal pada rasio 1x augmentasi dan menurun pada rasio 2x hingga 3x [19]. Di sisi lain, Hossain dan Zhang (2024) menunjukkan bahwa pada data MRI otak, peningkatan rasio augmentasi hingga 4x

secara konsisten meningkatkan *accuracy* klasifikasi menggunakan CNN dan Vision Transformer [20]. Tabel 2.9 merangkum hasil-hasil tersebut dan menunjukkan bahwa rasio augmentasi optimal bergantung pada jenis data dan metode augmentasi yang digunakan.

2.4 Data Splitting

Pembagian dataset atau data splitting merupakan langkah krusial dalam proses pelatihan model machine learning untuk memastikan kemampuan generalisasi model terhadap data baru. Dalam penelitian ini, akan dilakukan dua percobaan dengan rasio pembagian data yang berbeda, yaitu 80:20 dan 75:25, untuk mengevaluasi pengaruhnya terhadap performa model klasifikasi citra.

- ***Training set* (80% / 75%)**: Digunakan untuk melatih model agar dapat mempelajari pola dan relasi antar fitur terhadap label.
- ***Testing set* (20% / 25%)**: Digunakan untuk mengevaluasi performa model setelah pelatihan, dengan data yang tidak pernah dilihat sebelumnya oleh model.

Pemilihan rasio 80:20 didasarkan pada praktik umum yang memberikan keseimbangan antara jumlah data yang cukup untuk pelatihan dan evaluasi yang representatif. Beberapa penelitian mendukung efektivitas rasio ini, seperti dalam penelitian oleh Joseph (2022) yang menyatakan bahwa rasio 80:20 merupakan pilihan yang seimbang dan banyak digunakan dalam berbagai studi eksperimental[21], serta oleh Nguyen et al. (2021) yang menunjukkan bahwa pemilihan rasio pembagian data berpengaruh signifikan terhadap *accuracy* prediksi, dan rasio 80:20 memberikan performa terbaik pada penelitian yang dilakukan[22].

Namun, rasio 75:25 juga menunjukkan hasil yang kompetitif dalam beberapa studi. Penelitian oleh Oktafiani et al. (2023) menunjukkan bahwa penggunaan rasio 75:25 pada klasifikasi kanker payudara dengan algoritma *Support Vector Machine* (SVM) menghasilkan *accuracy* tertinggi sebesar 98.89% dibandingkan dengan rasio lainnya seperti 55:45 atau 80:20[23]. Selain itu, penelitian oleh Sivakumar et al. (2024) menekankan pentingnya keseimbangan antara data pelatihan dan pengujian untuk menghindari *overfitting* atau *underfitting*, rasio 75:25 memberikan performa yang lebih baik jika dibandingkan dengan rasio 80:20 pada klasifikasi citra medis[24].

2.5 InceptionV3

InceptionV3 adalah arsitektur jaringan saraf konvolusional (CNN) yang dikembangkan oleh Szegedy et al. [25] dan merupakan versi lanjut dari arsitektur *Inception* sebelumnya. Model ini dirancang untuk meningkatkan efisiensi komputasi dan *accuracy* klasifikasi dengan menggunakan modul *Inception*, yang memungkinkan ekstraksi fitur dari berbagai ukuran filter secara paralel. Dalam penelitian ini, model InceptionV3 yang digunakan diambil dari *library tensorflow.keras.applications*.

Penerapan InceptionV3 untuk ekstraksi fitur dilakukan dengan pengaturan sebagai berikut:

- *Weights* berfungsi untuk menentukan *pre trained model* yang akan digunakan.
- *include_top* berfungsi untuk *layer* klasifikasi.
- *GlobalAveragePooling2D* berfungsi sebagai *layer* output.
- *input_shape* berfungsi untuk mengatur ukuran input *image*.

2.5.1 Modul Inception

InceptionV3 menggunakan modul *Inception* yang terdiri dari beberapa jalur konvolusi dengan ukuran filter yang berbeda, yang memungkinkan model menangkap informasi dari berbagai skala secara simultan. Dengan pendekatan ini, model dapat menangkap fitur pada berbagai tingkat abstraksi tanpa meningkatkan jumlah parameter secara signifikan.

2.5.2 Struktur InceptionV3

InceptionV3 terdiri dari beberapa tahap konvolusi dan modul *Inception* yang dioptimalkan untuk efisiensi dan *accuracy* tinggi. Struktur utama InceptionV3 dapat dirangkum sebagai berikut:

- **Lapisan awal:** Konvolusi 3 × 3 dengan 32 filter, diikuti oleh konvolusi 3 × 3 dengan 64 filter, dan operasi *max pooling*.
- **Modul *Inception*:** Terdiri dari beberapa tahap dengan kombinasi filter konvolusi:

1. *Inception*-A: Menggunakan konvolusi 1 × 1, 3 × 3, dan 5 × 5 dengan berbagai jalur.
2. *Inception*-B: Menggunakan faktorisasi konvolusi untuk meningkatkan efisiensi, seperti konvolusi 7 × 7 yang dipecah menjadi dua konvolusi berturut-turut 1 × 7 dan 7 × 1.
3. *Inception*-C: Menggunakan faktorisasi lebih lanjut dengan konvolusi 1 × 3 dan 3 × 1 untuk menangkap lebih banyak variasi fitur.

– **Lapisan akhir:** *Global Average Pooling* (GAP), diikuti oleh lapisan *fully connected* dan fungsi aktivasi *softmax* untuk klasifikasi.

2.6 Bird Eye View Feature Selection

Bird Eye View Feature Selection adalah algoritma *feature selection* yang menggabungkan *Markov Chain*, *Evolutionary Algorithm*, dan *Reinforced Learning*[10]. Algoritma dimulai dari sebuah *root node* dan dari *node* tersebut akan menghasilkan *child node*. Proses ini akan berulang sampai *node* yang memiliki akurasi tertinggi didapatkan. Pada *root node* berisikan pasangan *subset* fitur yang dipilih acak dan direpresentasikan dengan angka 1 atau nol, dimana 1 berarti fitur tersebut dipilih dan begitu pula sebaliknya. Proses berlanjut kepada generasi *node* baru yang yang isinya telah tertransisi. Transisi tersebut diperoleh dari hasil probabilitas *Markov Chain*. Selama proses generasi *node* berlanjut, matriks transisi akan diperbaharui sesuai dengan *reward* dari keakurasian fitur-fitur yang ada di *node*.

Definisi berikut adalah penjelasan tentang pendekatan yang digunakan:

– States: $\{0,1\}^d$.

– $\mathscr{A}$: Jumlah *child node* yang dihasilkan oleh setiap *parent node*.

– $\mathscr{M}_{\mathscr{A}}$: Jumlah *node* yang dipilih berdasarkan tingkat performa tiap iterasi.

– t: Jumlah iterasi.

– s: Jumlah tahap.

– $\mathscr{F}_j^{t,s}$: Mewakili status *node* nomor j pada waktu t dan tahap s, $\mathscr{F}_j^{t,s} \in \{0,1\}^d$, $j = 1,...,\mathscr{M}_{\mathscr{A}}$, yang menentukan apakah setiap fitur telah dipilih atau tidak. Posisi dengan nilai 1 menunjukkan fitur yang telah dipilih, sedangkan posisi dengan nilai 0 menunjukkan fitur yang telah dieliminasi.

– $f_{i,j}^{t,s}$: Nilai fitur ke-i pada *node* nomor j pada waktu t dan tahap s, $f_{i,j}^{t,s} \in \{0,1\}$, dengan $i = 1,2,...,d$ dan $j = 1,...,\mathscr{M}_{\mathscr{A}}$.

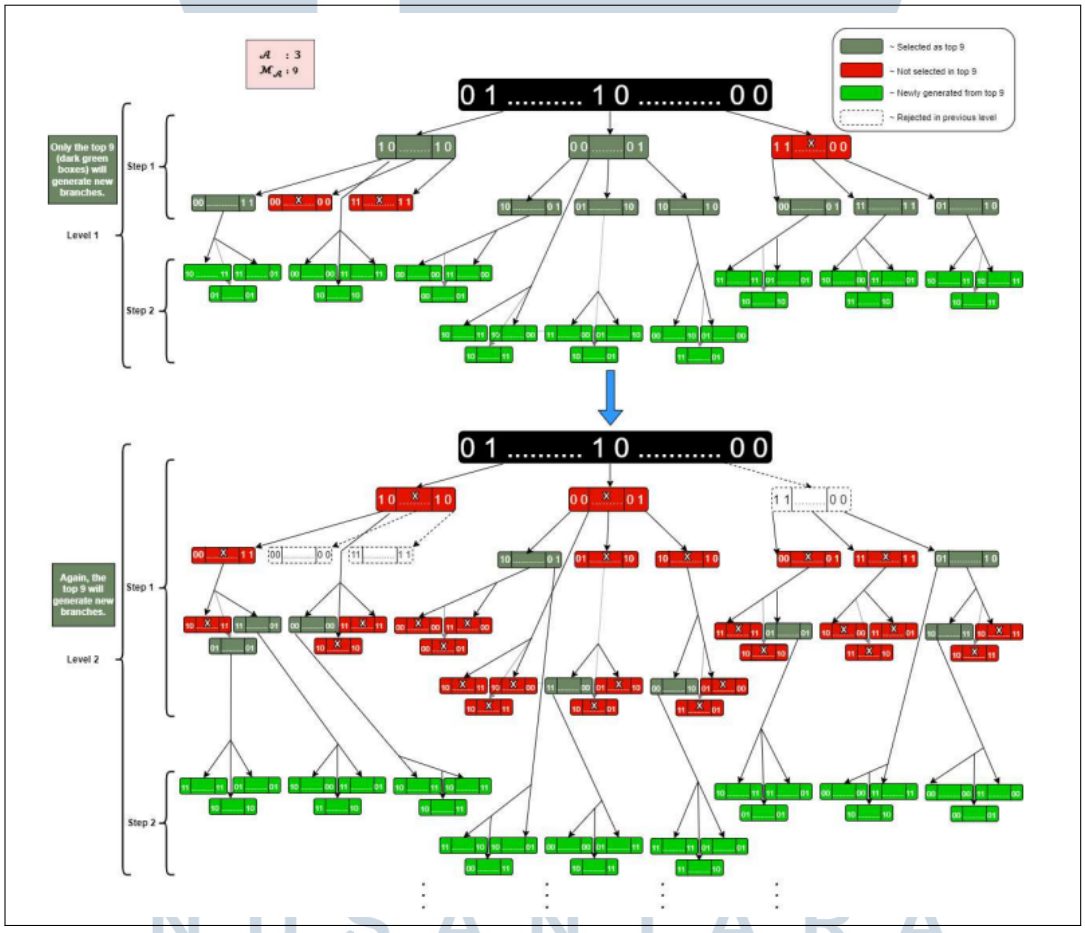
– $c_{i,j}^{t,s}$: *state* pasangan fitur ke-i, $c_{i,j}^{t,s} = \{f_{2i-1,j}^{t,s}, f_{2i,j}^{t,s}\}$, pada waktu t dan tahap s dari *node* ke-j.

– $p_{i,j}^{t-1,s}(c_{i,j}^{t,s}|c_{i,j}^{t-1,s})$: Probabilitas transisi dari pasangan fitur $c_{i,j}^{t-1,s}$ ke pasangan $c_{i,j}^{t,s}$, yang mewakili tindakan dari algoritma evolusi.

– d: Dimensi data atau jumlah fitur $f_{i,j}^{t,s}$, dengan $i = 1,2,...,d$.

– n: Jumlah observasi data.

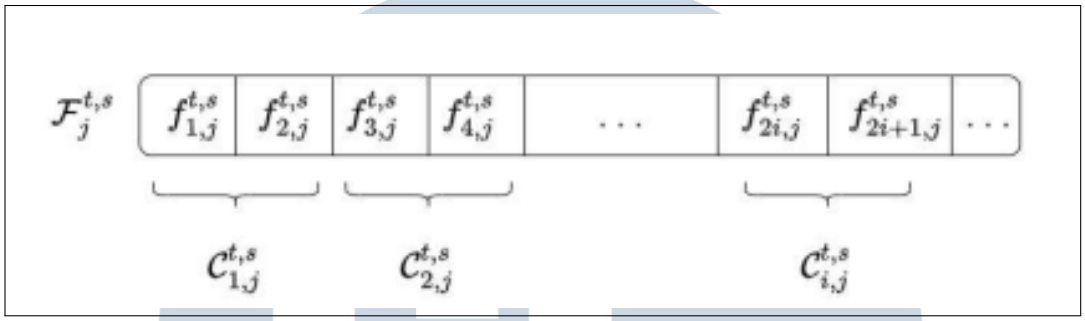
– ε: Fungsi reward.



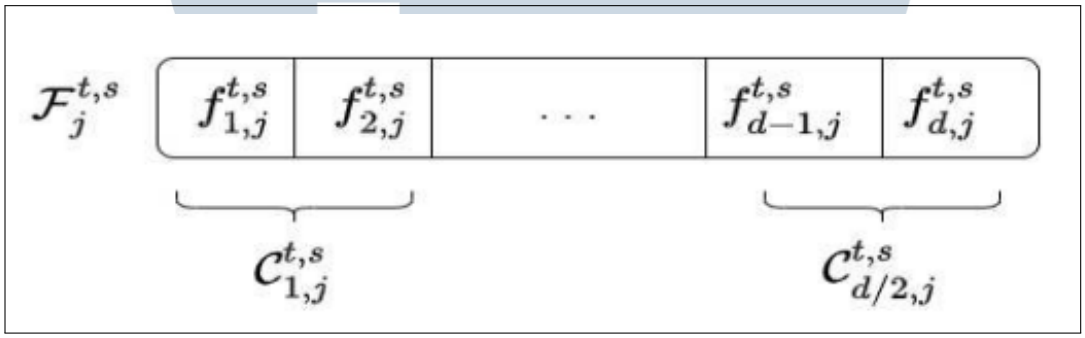
Gambar 2.1. Ilustrasi *tree* BEV

Gambar 2.1 menunjukkan proses dari BEV dengan $\mathscr{A}$ bernilai 3 dan $\mathscr{M}_{\mathscr{A}}$ bernilai 9. *Node* berwarna hitam merupakan kumpulan nilai pasangan *subset* fitur

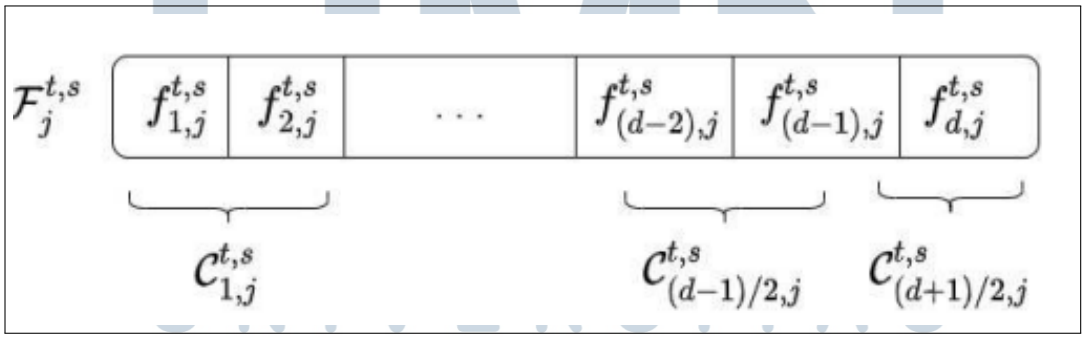
yang dipilih secara acak. *Root node* tersebut menghasilkan 3 *child note*. Lalu dari 3 *child node* tersebut digenerasikan 3 *node*. Setelah itu akan dipilih 9 *node* terbaik.



Gambar 2.2. *Node* yang berisikan pasangan fitur-fitur

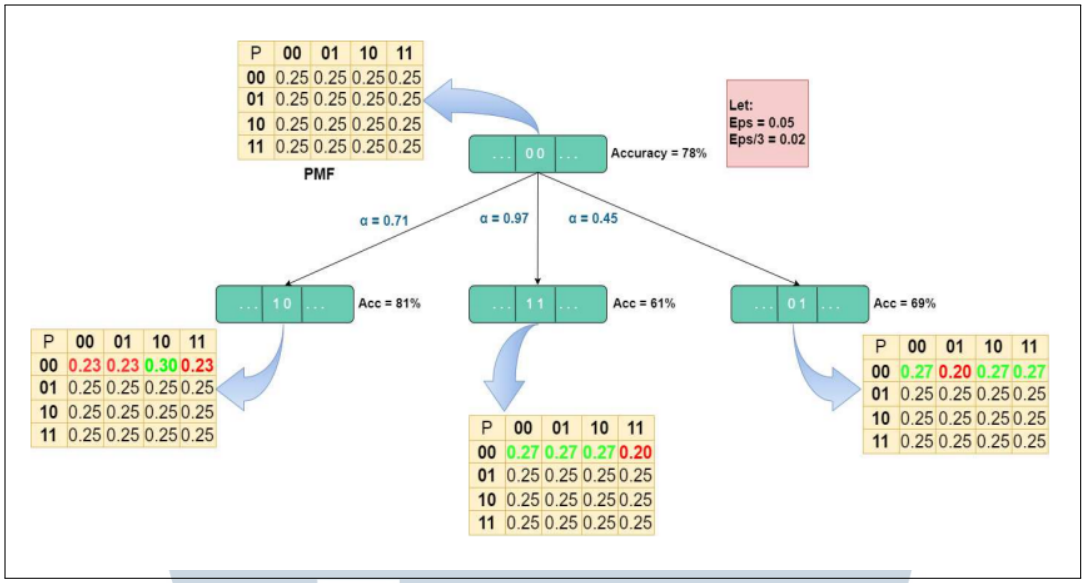


Gambar 2.3. Pasangan fitur jika jumlah fitur adalah genap



Gambar 2.4. Pasangan fitur jika jumlah fitur adalah ganjil

Setiap *node* berisikan pasangan fitur-fitur yang direpresentasikan dengan angka 1 dan 0 seperti Gambar 2.2. Setiap pasangan fitur memiliki matriks transisi masing-masing yang akan mempengaruhi *state* dari pasangan fitur tersebut. Gambar 2.3 menunjukkan pembagian pasangan jika jumlah fitur adalah genap dan Gambar 2.4 menunjukkan pembagian pasangan jika jumlah fitur adalah ganjil.



Gambar 2.5. Simulasi pembaharuan *state* pada setiap iterasi

$$\mathcal{C}_{i,(j-1)\mathcal{A}+r}^{t+1,s} = \begin{cases} \{0,0\} & if \quad \alpha_{i,j,r}^{t,s} < \mathcal{P}_{0,i,(j-1)\mathcal{A}+r}^{t,s} \\ \{0,1\} & if \quad \mathcal{P}_{0,i,(j-1)\mathcal{A}+r}^{t,s} \leq \alpha_{i,j,r}^{t,s} < \mathcal{P}_{0,i,(j-1)\mathcal{A}+r}^{t,s} + \mathcal{P}_{1,i,(j-1)\mathcal{A}+r}^{t,s} \\ \{1,0\} & if \quad \mathcal{P}_{0,i,(j-1)\mathcal{A}+r}^{t,s} + \mathcal{P}_{1,i,(j-1)\mathcal{A}+r}^{t,s} \leq \alpha_{i,j,r}^{t,s} < \mathcal{P}_{0,i,(j-1)\mathcal{A}+r}^{t,s} + \mathcal{P}_{1,i,(j-1)\mathcal{A}+r}^{t,s} + \mathcal{P}_{2,i,(j-1)\mathcal{A}+r}^{t,s} \\ \{1,1\} & Elsewhere \end{cases}$$

Gambar 2.6. Formula pembaharuan *state*

Gambar 2.5 mengilustasikan bagaimana *Markov Chain* dan *Reinforced Learning* bekerja dalam BEV. Setiap pasangan fitur memiliki matriks *Markov* tersendiri dan berdasarkan probabilitas beserta epsilon yang akan menentukan *state* dari pasangan berikutnya. Gambar 2.6 adalah aturan atau formula dari pembaharuan *state*.

2.7 XGBoost

XGBoost adalah algoritma *ensemble* yang menggunakan *gradient boosting* untuk meningkatkan akurasi prediksi dengan menambahkan pohon keputusan (*decision tree*) secara iteratif. Setiap pohon keputusan baru dibuat untuk memperbaiki kesalahan prediksi dari pohon sebelumnya dengan cara meminimalkan *loss function*[26]. Dalam penelitian ini, model XGBoost yang digunakan diambil dari *library xgboost*.

Penerapan XGBoost untuk klasifikasi dilakukan dengan pengaturan sebagai berikut:

- *objective* berfungsi untuk menentukan jenis klasifikasi multikelas atau regresi.
- *num_class* berfungsi untuk jumlah kelas untuk klasifikasi.
- *tree_method* berfungsi untuk menentukan metode pembentukan pohon.
- *device* berfungsi untuk menentukan perangkat keras yang digunakan.
- *max_depth* berfungsi untuk mengatur kedalaman maksimum setiap pohon.
- *min_child_weight* berfungsi untuk menentukan jumlah minimum bobot satu *leaf*.
- *learning_rate* berfungsi untuk mengontrol kecepatan pembelajaran.
- *n_estimators* berfungsi untuk menentukan jumlah maksimum pohon yang akan dibentuk.
- *subsample* berfungsi untuk menentukan rasio data pelatihan yang akan digunakan untuk membangun setiap pohon.
- *colsample_bytree* berfungsi untuk menentukan rasio fitur yang dipilih secara acak untuk setiap pohon.
- *gamma* berfungsi untuk menentukan nilai minimum untuk pemangkasan pohon atau *pruning*.
- *reg_alpha* berfungsi untuk parameter regularisasi L1 (*Lasso*).
- *reg_lambda* berfungsi untuk parameter regularisasi L2 (*Ridge*).
- *eval_metric* berfungsi untuk menentukan metrik evaluasi pohon.
- *early_stopping_rounds* berfungsi untuk menghentikan pelatihan secara otomatis jika tidak terjadi peningkatan pada evaluasi selama sejumlah iterasi tertentu.

2.8 Accuracy, precision, recall, F1 score

Evaluasi kinerja model klasifikasi dapat diukur menggunakan beberapa metrik, seperti *accuracy, precision, recall*, dan *F1 score*. Berikut adalah penjelasan masing-masing metrik.

2.8.1 Accuracy

Accuracy adalah metrik yang mengukur seberapa banyak prediksi yang benar dibandingkan dengan jumlah total sampel[27]. Didefinisikan seperti pada Persamaan 2.1:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.1}$$

Di mana:

- TP (*True Positive*) adalah jumlah sampel positif yang diprediksi benar.
- TN (*True Negative*) adalah jumlah sampel negatif yang diprediksi benar.
- FP (*False Positive*) adalah jumlah sampel negatif yang diprediksi sebagai positif.
- FN (*False Negative*) adalah jumlah sampel positif yang diprediksi sebagai negatif.

2.8.2 Precision

Precision mengukur proporsi sampel yang diprediksi positif yang benar-benar merupakan kelas positif[28]. *Precision* didefinisikan seperti pada Persamaan 2.2:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.2}$$

Semakin tinggi nilai *precision*, semakin sedikit prediksi positif yang salah.

2.8.3 Recall

Recall mengukur seberapa baik model dalam menemukan semua sampel positif dari keseluruhan data yang benar-benar positif[29]. Didefinisikan pada Persamaan 2.3:

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.3}$$

Nilai *recall* yang tinggi menunjukkan bahwa model dapat menangkap sebagian besar sampel positif yang ada.

2.8.4 F1 Score

F1 Score adalah rata-rata harmonik antara *precision* dan *recall*, yang memberikan keseimbangan antara keduanya[30]. Didefinisikan pada Persamaan 2.4:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

Metrik ini sangat berguna ketika terdapat ketidakseimbangan kelas, di mana *accuracy* saja tidak cukup untuk mengevaluasi performa model.