LAMPIRAN

Surat Pernyataan Publikasi Data LPKU ITN Malang

SURAT PERNYATAAN

Yang bertandatangan di bawah ini :

Nama	: Franciscus Xaverius Ariwibisono S.T., M.Kom.
Jabatan	: Sekretaris
Instansi/Lembaga	: Lembaga Pengembangan Kerjasama dan Usaha ITN Malang
No. Telepon/HP	: +62 853-3557-4770
Alamat	: Jl. Sigura - Gura No.2, Sumbersari, Kec. Lowokwaru, Kota Malang, Jawa Timur 65152

Dengan ini menyatakan bersedia menjadi lokasi penelitian dalam pembuatan skripsi :

Nama	: Nayura Arsineda
NIM	: 2218022
Judul Skripsi	: Sistem Pendukung Keputusan Analisis Risiko Kerjasama Kemitraan Perguruna Tinggi Menggunakan Metode Decision Tree

Dan saya menyatakan bahwa data yang digunakan dalam skripsi tersebut adalah benar dan boleh dipublikasikan. Demikian surat pernyataan ini saya buat tanpa ada paksaan dan tekanan dari pihak manapun serta dapat dipergunakan sebagaimana mestinya.

Catatan :
Nama mitra dgn jabatan / nama lain.

Malang, 1 Juni 2026
Yang membuat pernyataan

(F.X. ARIWIBISONO, S.T., M.Kom.)

Formulir Bimbingan Tugas Akhir 1

FORMULIR BIMBINGAN TUGAS AKHIR

Nama : Nayura Arfineda

Nim : 2218022

Masa Bimbingan : 21 Februari s/d 26 Agustus 2026

Judul tugas akhir : Sistem Pendukung Keputusan Analisis Risiko Kerjasama Kemitraan Perguruan Tinggi Menggunakan Metode Decision Tree

No.	Tanggal	Uraian	Paraf Pembimbing
1.	07-04-2026	konsultasi mengenai website spk.	
2.	14-04-2026	Bimbingan mengenai atribut/parameter spk.	
3.	22-04-2026	Bimbingan mengenai website	
4.	5-05-2026	konsultasi mengenai atribut/parameter	
5.	11-05-2026	presentasi hasil website, perubahan	
6.		dari localhost ke server Lpku.	
7.	18-05-2026	Laporan perkembangan web.	
8.	25-05-2026	Mengenai kelengkapan data	
9.			
10.			

Malang,..... 2026

Dosen Pembimbing

F.X. Ariwibisono S.T. M.Kom.

Formulir Bimbingan Tugas Akhir 2

INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus 1 : Jln. Bendungan Sigura-Gura No.2 Malang
Kampus 2 : Jln. Raya Karanglo Km.2 Malang

FORMULIR BIMBINGAN TUGAS AKHIR

Tahun Akademik	:	2025 / Genap
Mahasiswa	:	2218022 - NAYURA ARSINEDA
Prodi	:	Teknik Informatika S-1
Dosen Pembimbing	:	1115 - NURLAILY VENDYANSYAH, ST., MT
Judul	:	Sistem Pendukung Keputusan Analisis Risiko Kerjasama Kemitraan Perguruan Tinggi Menggunakan Metode Decision Tree

Pertemuan	Tanggal	Uraian	Paraf
1	09-04-2026	Bab 2 algoritma harus ada persamaan matematis berikut step by step penejelasannya. Bab 3 peracangan metode C4.5 yang sudah disesuaikan dengan project.	
2	16-04-2026	Konsultasi format penulisan laporan tugas akhir dan cara menulis sesuai kebutuhan pennelitian.	
3	22-04-2026	Persamaan matematika, tree, tabel dataset. Bab 2 berisi teori umum (bukan penelitian Anda). Perancangan bab 3 hanya menggunakan persamaan di bab 2. Untuk DFD tidak perlu. Data LPKU harus sudah ada.	
4	07-05-2026	Diskusi hasil seminar progres, Next week : Demo final project, Metode belum implemented gpp (belum berfungsi gpp), merencanakan item uji Pengujian blackbox, pengujian responden (tim LPKU).	
5	21-05-2026	Data harus sudah lengkap. Jika data tidak lengkap maka Anda tidak akan dapat melanjutkann ke tahap seminar hasil. Anda tidak akan bisa menulis kesimpulan dan pengujian. Deadline senin, 25 Mei 2026.	
6	26-05-2026	Datasheet belum diperoleh dari LPKU (masih dijanjikan terus menerus) sehingga tidak bisa dilakukan pengujian	
7	29-05-2026	Masih sama. Datasheet masih dijanjikan.	

Malang, 29-05-2026

Dosen Pembimbing

NURLAILY VENDYANSYAH, ST., MT

Lampiran Dosen Pembimbing 1

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang
PT. BNI (PERSERO) MALANG
BANK NAGA MALANG

Malang, 02 April 2026

Nomor : ITN-169/III.INF/TA/2026
Lampiran : ---
Perihal : *Pembimbing Utama Tugas Akhir*

Kepada : Yth. Bapak/ibu FX.Ariwibisono, ST., M.Kom.
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal tugas akhir untuk mahasiswa :

Nama : Nayura Arsineda
Nim : 2218022
Prodi : Teknik Informatika S-1
Fakultas : Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

21 Februari 2026 s/d 21 Agustus 2026

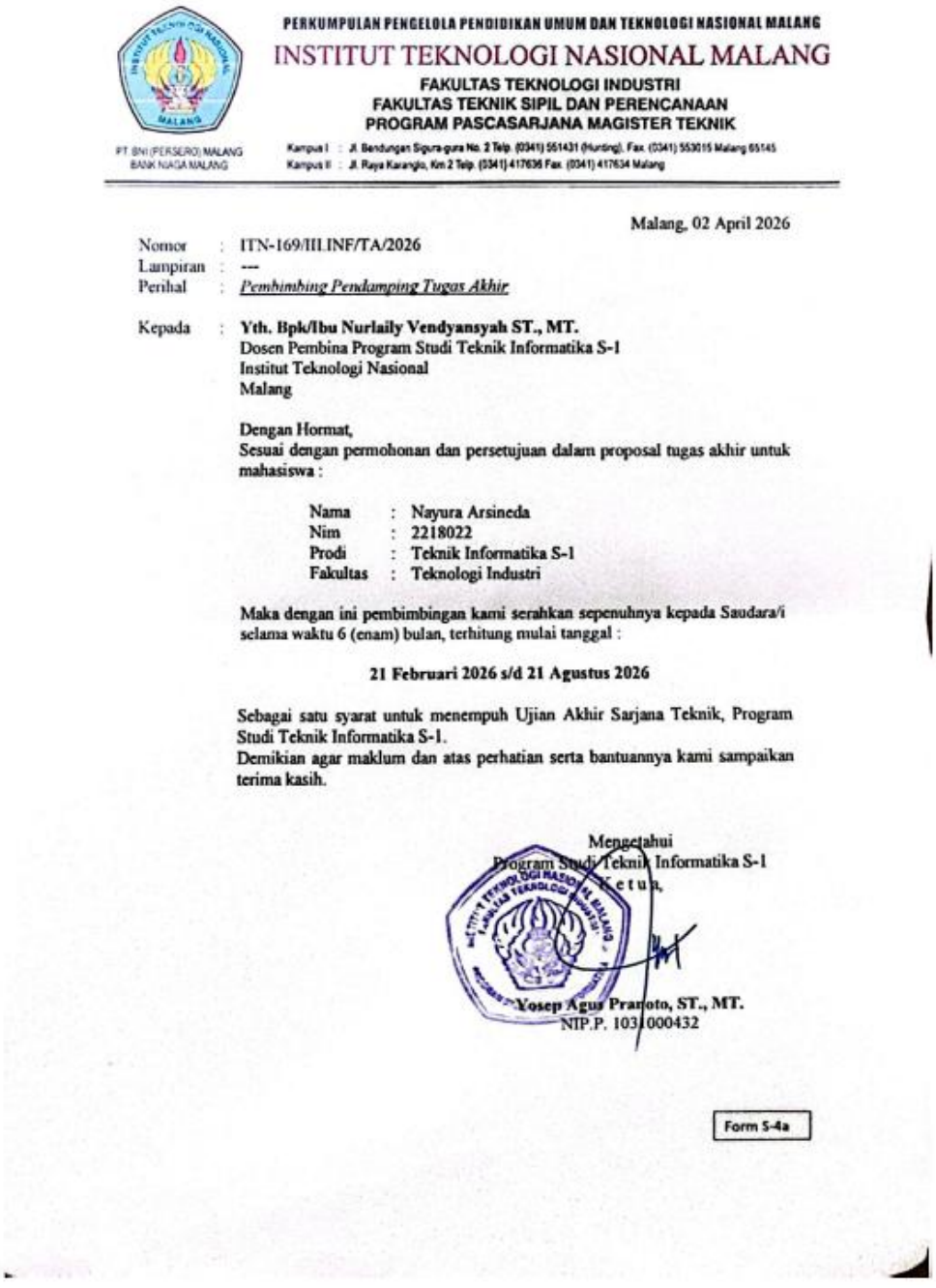
Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
Ketua,

Yosep Agus Pranoto, ST., MT.
NIP.P. 103 000432

Form S-4a

Lampiran Dosen Pembimbing 2



PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 02 April 2026

Nomor : ITN-169/III.INF/TA/2026
Lampiran : ---
Perihal : *Pembimbing Pendamping Tugas Akhir*

Kepada : **Yth. Bpk/Ibu Nurlaily Vendyansyah ST., MT.**
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal tugas akhir untuk mahasiswa :

Nama	:	Nayura Arsineda
Nim	:	2218022
Prodi	:	Teknik Informatika S-1
Fakultas	:	Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

21 Februari 2026 s/d 21 Agustus 2026

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
Ketua,

Yosep Agus Pranoto, ST., MT.
NIP.P. 1031000432

Form S-4a

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK MAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN TUGAS AKHIR

FAKULTAS TEKNOLOGI INDUSTRI

Nama : **Nayura Arsineda**
Nim : **2218022**
Jurusan : **Teknik Informatika S-1**
JUDUL : **SISTEM PENDUKUNG KEPUTUSAN ANALISIS RISIKO KERJASAMA KEMITRAAN PERGURUAN TINGGI MENGGUNAKAN METODE DECISION TREE**

Dipertahankan Dihadapan Majelis Penguji Tugas Akhir Jenjang Strata Satu(S-1) Pada

Hari : **Senin**
Tanggal : **06 Juli 2026**
Nilai : **88**

Panitia Ujian Tugas Akhir
Penguji I

Ahmad Faisol S.T., M.T.
NIP.P 1031000431

Panitia Ujian Tugas Akhir
Penguji II

Hani Zulfia Zahro, S. Kom. M.Kom.
NIP.P. 1031500480

Panitia Ujian Tugas Akhir
Ketua Majelis Penguji

Yosep Agus Pranoto, ST, MT
NIP.P. 1031000432

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

FORMULIR PERBAIKAN TUGAS AKHIR

Dalam pelaksanaan ujian Tugas Akhir jenjang Strata 1 Program Studi Teknik Informatika, maka perlu adanya perbaikan Tugas Akhir untuk mahasiswa :

NAMA : Nayura Arsineda
NIM : 2218022
JURUSAN : Teknik Informatika S-1
JUDUL : Sistem Pendukung Keputusan Analisis Risiko Kerjasama Kemitraan Perguruan Tinggi Menggunakan Metode Decision Tree

No.	Penguji	Tanggal	Uraian	Paraf
1.	Penguji I	07 Juli 2026	1. Validasi form MOU 2. Validasi file pdf 3. Validasi file historis Excel jika tidak sesuai format	
2.	Penguji II	07 Juli 2026	1. Script di lampiran 2. Revisi program 3. Revisi laporan 4. Perhitungan	

Dosen Penguji I

(Ahmad Faisol, S.T., M.T.)
NIP.P 1031000431

Dosen Penguji II

(Hani Zulfia Zahro', S. Kom. M.Kom.)
NIP.P. 1031500480

Dosen Pembimbing I

(Franciscus Xaverius Ariwibisono ST., M.Kom.)
NIP.P. 1030300397

Dosen Pembimbing II

(Nurlaily Vendyansyah, S.T., M.T.)
NIP.P. 1031900557

INSTITUT TEKNOLOGI NASIONAL MALANG
PERPUSTAKAAN PUSAT
Jln. Bendungan Sigura-gura No.2 Malang 65145
Telp. (0341) 551431 Pes. 163-146-147 Fax. (0341) 553015 Website : library.itn.ac.id

FORM UJI PLAGIASI UNTUK MAHASISWA

Yang bertandatangan di bawah ini, Mahasiswa Institut Teknologi Nasional Malang:

Nama	: Nayora Arrneda
NIM	: 22180022
Fakultas / Jurusan	: Fakultas Teknologi Industri / Teknik Informatika S1
Email	: 22180022@scholar.itn.ac.id
No. Tlp	: 081219072423
Judul/ Jml artikel	: Sistem Pendukung Keputusan Analisis Risiko Kerja Jarak Kemiraan Perguruan Tinggi Menggunakan Metode Decision Tree

Karya ilmiah yang bersangkutan di atas melalui proses cek plagiatsi menggunakan aplikasi trunitin dengan hasil kemiripan (Similarty) Sebesar…….. %
Demikian surat keterangan ini dibuat agar dapat dipergunakan sebagaimana mestinya.

Malang, 3 Agustus 2024

Mahasiswa

Nayora Arrneda

Pelaksana,

Handoyo Eka

Laporan Tugas 2218022 Nayura Arsineda - NAYURA ARSINEDA.pdf

ORIGINALITY REPORT

2%
SIMILARITY INDEX

PRIMARY SOURCES

1	eprints.itn.ac.id Internet	159 words — 1%
2	pdfs.semanticscholar.org Internet	84 words — 1%

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	< 1%
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF

Source Code Pohon Keputusan

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;

class c45PohonKeputusanAdminController extends Controller
{
    protected string $tableDataset = 'c45data_historis';
    protected string $tablePohon = 'c45pohon_keputusan';

    protected array $atribut = [
        'kelengkapan_dokumen'  => 'C1 - Kelengkapan Dokumen
Kerja Sama',
        'tingkat_implementasi' => 'C2 - Tingkat Implementasi
Kerja Sama',
        'cakupan_aktivitas'    => 'C3 - Cakupan Aktivitas Kerja
Sama',
        'durasi_kerjasama'     => 'C4 - Durasi Kerja Sama',
    ];

    protected string $target = 'tingkat_risiko';

    protected array $kelas = [
        'Rendah',
        'Sedang',
        'Tinggi',
    ];

    protected string $tipeDataTraining = 'Training';

    /**
     * Mengambil data training dan membangun pohon keputusan
C4.5.
     */
    public function generate()
    {
        try {
            DB::beginTransaction();

            DB::table($this->tablePohon)->delete();

            $datasets = DB::table($this->tableDataset)
                ->select(
                    'kelengkapan_dokumen',
                    'tingkat_implementasi',
                    'cakupan_aktivitas',
                    'durasi_kerjasama',
                    'tingkat_risiko'
                )
                ->where('tipe_data', $this->tipeDataTraining)
                ->whereNotNull('kelengkapan_dokumen')
                ->whereNotNull('tingkat_implementasi')
                ->whereNotNull('cakupan_aktivitas')
                ->whereNotNull('durasi_kerjasama')
```

```
                ->whereNotNull('tingkat_risiko')
                ->whereIn('tingkat_risiko', $this->kelas)
                ->get()
                ->map(fn ($item) => (array) $item)
                ->toArray();

            if (count($datasets) === 0) {
                DB::rollBack();

                return redirect()
                    ->route('admin.analisis-risiko.mesin.pohon-
keputusan')
                    ->with(
                        'error',
                        'Data training masih kosong atau belum
lengkap.'
                    );
            }

            $this->buildTree(
                data: $datasets,
                atributTersedia: array_keys($this->atribut),
                parentPath: 'ROOT',
                kondisi: null,
                isRoot: true
            );

            DB::commit();

            return redirect()
                ->route('admin.analisis-risiko.mesin.pohon-
keputusan')
                ->with(
                    'success',
                    'Pohon keputusan C4.5 berhasil digenerate.'
                );
        } catch (\Exception $e) {
            if (DB::transactionLevel() > 0) {
                DB::rollBack();
            }

            Log::error(
                'GENERATE POHON KEPUTUSAN ERROR: ' . $e-
>getMessage()
            );

            return back()->with(
                'error',
                'Gagal generate pohon keputusan: ' . $e-
>getMessage()
            );
        }
    }

    /**
     * Membangun pohon keputusan secara rekursif.
     */
    private function buildTree(
        array $data,
```

```
        array $atributTersedia,
        string $parentPath = 'ROOT',
        ?string $kondisi = null,
        bool $isRoot = false
    ): void {
        $jumlahData = count($data);

        if ($jumlahData === 0) {
            return;
        }

        $jumlahRendah = $this->countTarget($data, 'Rendah');
        $jumlahSedang = $this->countTarget($data, 'Sedang');
        $jumlahTinggi = $this->countTarget($data, 'Tinggi');
        $entropy = $this->entropy($data);

        if ($this->isPure($data) || empty($atributTersedia)) {
            $this->insertLeaf(
                $data,
                $parentPath,
                $kondisi,
                $entropy,
                $isRoot
            );

            return;
        }

        $best = $this->bestAttribute($data, $atributTersedia);

        if (!$best || $best['gain'] <= 0) {
            $this->insertLeaf(
                $data,
                $parentPath,
                $kondisi,
                $entropy,
                $isRoot
            );

            return;
        }

        $bestAtribut = $best['atribut'];
        $bestGain = $best['gain'];

        DB::table($this->tablePohon)->insert([
            'atribut'        => $bestAtribut,
            'kondisi'        => $kondisi,
            'parent'         => $parentPath,
            'jumlah_data'    => $jumlahData,
            'jumlah_rendah'  => $jumlahRendah,
            'jumlah_sedang'  => $jumlahSedang,
            'jumlah_tinggi'  => $jumlahTinggi,
            'entropy'        => $entropy,
            'gain'           => $bestGain,
            'keputusan'      => null,
            'is_root'        => $isRoot ? 1 : 0,
            'is_leaf'        => 0,
            'created_at'     => now(),
```

```
            'updated_at'     => now(),
        ]);

        $nilaiAtribut = collect($data)
            ->pluck($bestAtribut)
            ->filter()
            ->unique()
            ->values()
            ->toArray();

        $sisaAtribut = array_values(array_filter(
            $atributTersedia,
            fn ($atribut) => $atribut !== $bestAtribut
        ));

        foreach ($nilaiAtribut as $nilai) {
            $subset = array_values(array_filter(
                $data,
                fn ($row) => ($row[$bestAtribut] ?? null) ===
$nilai
            ));

            if (empty($subset)) {
                continue;
            }

            $childPath = $parentPath
                . '|'
                . $bestAtribut
                . '='
                . $nilai;

            $this->buildTree(
                data: $subset,
                atributTersedia: $sisaAtribut,
                parentPath: $childPath,
                kondisi: $nilai,
                isRoot: false
            );
        }
    }

    /**
     * Menyimpan node daun beserta keputusan kelas mayoritas.
     */
    private function insertLeaf(
        array $data,
        string $parentPath,
        ?string $kondisi,
        float $entropy,
        bool $isRoot = false
    ): void {
        DB::table($this->tablePohon)->insert([
            'atribut'        => null,
            'kondisi'        => $kondisi,
            'parent'         => $parentPath,
            'jumlah_data'    => count($data),
            'jumlah_rendah'  => $this->countTarget($data,
'Rendah'),
```

```
            'jumlah_sedang'  => $this->countTarget($data,
'Sedang'),
            'jumlah_tinggi'  => $this->countTarget($data,
'Tinggi'),
            'entropy'        => $entropy,
            'gain'           => null,
            'keputusan'      => $this->majorityClass($data),
            'is_root'        => $isRoot ? 1 : 0,
            'is_leaf'        => 1,
            'created_at'     => now(),
            'updated_at'     => now(),
        ]);
    }

    /**
     * Menentukan atribut dengan nilai information gain
tertinggi.
     */
    private function bestAttribute(
        array $data,
        array $atributTersedia
    ): ?array {
        $bestAtribut = null;
        $bestGain = -1;

        foreach ($atributTersedia as $atribut) {
            $detail = $this->gainDetail($data, $atribut);
            $gain = $detail['gain'];

            if ($gain > $bestGain) {
                $bestGain = $gain;
                $bestAtribut = $atribut;
            }
        }

        if ($bestAtribut === null) {
            return null;
        }

        return [
            'atribut' => $bestAtribut,
            'gain'    => round($bestGain, 6),
        ];
    }

    /**
     * Menghitung entropy dan information gain suatu atribut.
     */
    private function gainDetail(
        array $data,
        string $atribut
    ): array {
        $total = count($data);

        if ($total === 0) {
            return [
                'atribut'          => $atribut,
                'entropy_total'    => 0,
                'weighted_entropy' => 0,
```

```
                'gain'              => 0,
            ];
        }

        $entropyTotal = $this->entropy($data);

        $nilaiAtribut = collect($data)
            ->pluck($atribut)
            ->filter()
            ->unique()
            ->values()
            ->toArray();

        $weightedEntropy = 0;

        foreach ($nilaiAtribut as $nilai) {
            $subset = array_values(array_filter(
                $data,
                fn ($row) => ($row[$atribut] ?? null) === $nilai
            ));

            $jumlahSubset = count($subset);

            if ($jumlahSubset === 0) {
                continue;
            }

            $entropySubset = $this->entropy($subset);
            $weight = $jumlahSubset / $total;

            $weightedEntropy += $weight * $entropySubset;
        }

        $gain = $entropyTotal - $weightedEntropy;

        return [
            'atribut'          => $atribut,
            'entropy_total'    => round($entropyTotal, 6),
            'weighted_entropy' => round($weightedEntropy, 6),
            'gain'             => round($gain, 6),
        ];
    }

    /**
     * Menghitung entropy berdasarkan distribusi kelas risiko.
     */
    private function entropy(array $data): float
    {
        $total = count($data);

        if ($total === 0) {
            return 0;
        }

        $entropy = 0;

        foreach ($this->kelas as $kelas) {
            $jumlah = $this->countTarget($data, $kelas);
```

```
            if ($jumlah === 0) {
                continue;
            }

            $probabilitas = $jumlah / $total;

            $entropy -= $probabilitas * log($probabilitas, 2);
        }

        return round($entropy, 6);
    }

    /**
     * Menghitung jumlah data pada kelas target tertentu.
     */
    private function countTarget(
        array $data,
        string $kelas
    ): int {
        return count(array_filter(
            $data,
            fn ($row) =>
                ($row[$this->target] ?? null) === $kelas
        ));
    }

    /**
     * Memeriksa apakah data hanya memiliki satu kelas target.
     */
    private function isPure(array $data): bool
    {
        return collect($data)
            ->pluck($this->target)
            ->filter()
            ->unique()
            ->count() === 1;
    }

    /**
     * Menentukan kelas mayoritas pada sebuah subset data.
     */
    private function majorityClass(array $data): string
    {
        $counts = [
            'Rendah' => $this->countTarget($data, 'Rendah'),
            'Sedang' => $this->countTarget($data, 'Sedang'),
            'Tinggi' => $this->countTarget($data, 'Tinggi'),
        ];

        arsort($counts);

        return array_key_first($counts) ?: 'Sedang';
    }

    /**
     * Melakukan prediksi dengan menelusuri pohon keputusan.
     */
    public function predict(array $row): string
    {
```

```
        $root = DB::table($this->tablePohon)
            ->where('is_root', 1)
            ->first();

        if (!$root) {
            return 'Sedang';
        }

        $node = $root;
        $path = 'ROOT';

        while ($node && (int) $node->is_leaf !== 1) {
            $atribut = $node->atribut;
            $nilai = $row[$atribut] ?? null;

            if (!$atribut || !$nilai) {
                return $this->fallbackDecisionFromNode($node);
            }

            $path .= '|' . $atribut . '=' . $nilai;

            $child = DB::table($this->tablePohon)
                ->where('parent', $path)
                ->orderBy('id_pohon', 'ASC')
                ->first();

            if (!$child) {
                return $this->fallbackDecisionFromNode($node);
            }

            $node = $child;
        }

        $decision = $node->keputusan
            ?? $this->fallbackDecisionFromNode($node);

        return in_array($decision, $this->kelas, true)
            ? $decision
            : 'Sedang';
    }

    /**
     * Menentukan keputusan cadangan berdasarkan kelas
terbanyak.
     */
    private function fallbackDecisionFromNode($node): string
    {
        if (!$node) {
            return 'Sedang';
        }

        $counts = [
            'Rendah' => (int) ($node->jumlah_rendah ?? 0),
            'Sedang' => (int) ($node->jumlah_sedang ?? 0),
            'Tinggi' => (int) ($node->jumlah_tinggi ?? 0),
        ];

        arsort($counts);
```

```
        $decision = array_key_first($counts) ?: 'Sedang';

        return in_array($decision, $this->kelas, true)
            ? $decision
            : 'Sedang';
    }
}
```

Source Code Confussion Matrix

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\c45DataHistoris;
use Illuminate\Support\Facades\DB;

class c45PengujianModelAdminController extends Controller
{
    protected string $tablePohon = 'c45pohon_keputusan';

    protected array $classes = [
        'Rendah',
        'Sedang',
        'Tinggi',
    ];

    protected string $tipeDataTraining = 'Training';
    protected string $tipeDataTesting = 'Testing';

    /**
     * Melakukan pengujian model C4.5 menggunakan data testing.
     */
    public function generate()
    {
        $root = DB::table($this->tablePohon)
            ->where('is_root', 1)
            ->first();

        if (!$root) {
            return redirect()
                ->route('admin.analisis-risiko.testing')
                ->with(
                    'error',
                    'Pohon keputusan belum digenerate.'
                );
        }

        $datasets = c45DataHistoris::where(
                'tipe_data',
                $this->tipeDataTesting
            )
            ->whereNotNull('kelengkapan_dokumen')
            ->whereNotNull('tingkat_implementasi')
            ->whereNotNull('cakupan_aktivitas')
            ->whereNotNull('durasi_kerjasama')
```

```
            ->whereNotNull('tingkat_risiko')
            ->whereIn('tingkat_risiko', $this->classes)
            ->get();

        $totalData = $datasets->count();

        if ($totalData <= 0) {
            return redirect()
                ->route('admin.analisis-risiko.testing')
                ->with(
                    'error',
                    'Data testing masih kosong.'
                );
        }

        $matrix = $this->emptyMatrix();
        $correct = 0;

        foreach ($datasets as $data) {
            $actual = $data->tingkat_risiko;

            $prediction = $this->predict([
                'kelengkapan_dokumen' =>
                    $data->kelengkapan_dokumen,

                'tingkat_implementasi' =>
                    $data->tingkat_implementasi,

                'cakupan_aktivitas' =>
                    $data->cakupan_aktivitas,

                'durasi_kerjasama' =>
                    $data->durasi_kerjasama,
            ]);

            if (!in_array(
                $prediction,
                $this->classes,
                true
            )) {
                $prediction = 'Sedang';
            }

            $matrix[$actual][$prediction]++;

            if ($actual === $prediction) {
                $correct++;
            }
        }

        $wrong = max($totalData - $correct, 0);

        $accuracy = (
            $correct / max($totalData, 1)
        ) * 100;

        $metrics = $this->calculateMetrics(
            $matrix,
            $totalData
```

```
        );

        return [
            'confusion_matrix' => $matrix,
            'total_testing' => $totalData,
            'correct_prediction' => $correct,
            'wrong_prediction' => $wrong,
            'accuracy' => round($accuracy, 2),
            'macro_precision' => round(
                $metrics['macro_precision'],
                2
            ),
            'macro_recall' => round(
                $metrics['macro_recall'],
                2
            ),
            'macro_f1_score' => round(
                $metrics['macro_f1_score'],
                2
            ),
        ];
    }

    /**
     * Membuat confusion matrix 3x3.
     */
    private function emptyMatrix(): array
    {
        return [
            'Rendah' => [
                'Rendah' => 0,
                'Sedang' => 0,
                'Tinggi' => 0,
            ],
            'Sedang' => [
                'Rendah' => 0,
                'Sedang' => 0,
                'Tinggi' => 0,
            ],
            'Tinggi' => [
                'Rendah' => 0,
                'Sedang' => 0,
                'Tinggi' => 0,
            ],
        ];
    }

    /**
     * Menghitung precision, recall, dan F1-score
     * untuk setiap kelas dan nilai macro average.
     */
    private function calculateMetrics(
        array $matrix,
        int $totalData
    ): array {
        $perClass = [];
        $precisionTotal = 0;
        $recallTotal = 0;
        $f1Total = 0;
```

```
        foreach ($this->classes as $class) {
            $tp = (int) $matrix[$class][$class];

            $fp = 0;

            foreach ($this->classes as $actualClass) {
                if ($actualClass !== $class) {
                    $fp += (int)
                        $matrix[$actualClass][$class];
                }
            }

            $fn = 0;

            foreach ($this->classes as $predictedClass) {
                if ($predictedClass !== $class) {
                    $fn += (int)
                        $matrix[$class][$predictedClass];
                }
            }

            $tn = max(
                $totalData - $tp - $fp - $fn,
                0
            );

            $precision = (
                $tp / max($tp + $fp, 1)
            ) * 100;

            $recall = (
                $tp / max($tp + $fn, 1)
            ) * 100;

            $f1 = (
                2 * ($precision * $recall)
            ) / max($precision + $recall, 1);

            $perClass[$class] = [
                'tp' => $tp,
                'tn' => $tn,
                'fp' => $fp,
                'fn' => $fn,
                'precision' => $precision,
                'recall' => $recall,
                'f1_score' => $f1,
            ];

            $precisionTotal += $precision;
            $recallTotal += $recall;
            $f1Total += $f1;
        }

        return [
            'per_class' => $perClass,

            'macro_precision' =>
                $precisionTotal
```

```
                / count($this->classes),

            'macro_recall' =>
                $recallTotal
                / count($this->classes),

            'macro_f1_score' =>
                $f1Total
                / count($this->classes),
        ];
    }

    /**
     * Melakukan prediksi dengan menelusuri
     * node pada pohon keputusan.
     */
    private function predict(array $row): string
    {
        $root = DB::table($this->tablePohon)
            ->where('is_root', 1)
            ->first();

        if (!$root) {
            return 'Sedang';
        }

        $node = $root;
        $path = 'ROOT';

        while (
            $node
            && (int) $node->is_leaf !== 1
        ) {
            $atribut = $node->atribut;
            $nilai = $row[$atribut] ?? null;

            if (!$atribut || !$nilai) {
                return $this
                    ->fallbackDecisionFromNode($node);
            }

            $path .= '|'
                . $atribut
                . '='
                . $nilai;

            $child = DB::table($this->tablePohon)
                ->where('parent', $path)
                ->orderBy('id_pohon', 'ASC')
                ->first();

            if (!$child) {
                return $this
                    ->fallbackDecisionFromNode($node);
            }

            $node = $child;
        }
```

```
        $decision = $node->keputusan
            ?? $this->fallbackDecisionFromNode($node);

        return in_array(
            $decision,
            $this->classes,
            true
        )
            ? $decision
            : 'Sedang';
    }

    /**
     * Menentukan keputusan cadangan berdasarkan
     * kelas mayoritas pada node terakhir.
     */
    private function fallbackDecisionFromNode(
        $node
    ): string {
        if (!$node) {
            return 'Sedang';
        }

        $counts = [
            'Rendah' =>
                (int) ($node->jumlah_rendah ?? 0),

            'Sedang' =>
                (int) ($node->jumlah_sedang ?? 0),

            'Tinggi' =>
                (int) ($node->jumlah_tinggi ?? 0),
        ];

        arsort($counts);

        $decision = array_key_first($counts)
            ?: 'Sedang';

        return in_array(
            $decision,
            $this->classes,
            true
        )
            ? $decision
            : 'Sedang';
    }
}
```

Source Code Analisis Risiko

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\c45ManajemenMitra;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
```

```
use Illuminate\Support\Facades\Validator;

class AnalisisRisikoAdminController extends Controller
{
    protected string $tablePohon = 'c45pohon_keputusan';

    /**
     * Memvalidasi, menghitung indikator, menentukan risiko,
     * dan menyimpan hasil analisis mitra.
     */
    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'mitra_id' => 'required',
            'sumber_data' => 'required|in:MoU,PKS',
            'nama_instansi' => 'nullable|string|max:255',
            'jumlah_mou' => 'nullable|integer|min:0',
            'jumlah_moa' => 'nullable|integer|min:0',
            'jumlah_ia' => 'nullable|integer|min:0',
            'jumlah_kegiatan' => 'nullable|integer|min:0',
            'jumlah_jenis_kegiatan' => 'nullable|integer|min:0',
        ]);

        if ($validator->fails()) {
            return back()
                ->withErrors($validator)
                ->withInput();
        }

        DB::beginTransaction();

        try {
            $type = $request->sumber_data;
            $table = strtolower($type) === 'mou'
                ? 'mou'
                : 'pks';

            $mitraId = $request->mitra_id;

            $mitra = DB::table($table)
                ->where('id', $mitraId)
                ->first();

            if (!$mitra) {
                DB::rollBack();

                return back()
                    ->with('error', 'Data mitra tidak
ditemukan!')
                    ->withInput();
            }

            $durasiTahun = $this->hitungDurasiTahun(
                $mitra->tanggal_mulai ?? null,
                $mitra->tanggal_berakhir ?? null
            );

            $jumlahMou = (int) ($request->jumlah_mou ?? 0);
            $jumlahMoa = (int) ($request->jumlah_moa ?? 0);
```

```
            $jumlahIa = (int) ($request->jumlah_ia ?? 0);
            $jumlahKegiatan = (int) ($request->jumlah_kegiatan
?? 0);
            $jumlahJenisKegiatan =
                (int) ($request->jumlah_jenis_kegiatan ?? 0);

            $kelengkapanDokumen =
                $this->hitungKelengkapanDokumen(
                    $jumlahMou,
                    $jumlahMoa,
                    $jumlahIa
                );

            $tingkatImplementasi =
                $this->hitungTingkatImplementasi(
                    $jumlahKegiatan
                );

            $cakupanAktivitas =
                $this->hitungCakupanAktivitas(
                    $jumlahJenisKegiatan
                );

            $durasiKerjasama =
                $this->hitungDurasiKerjasama(
                    $durasiTahun
                );

            $targetRisiko = $this->predictOrTemporary([
                'kelengkapan_dokumen' =>
                    $kelengkapanDokumen,

                'tingkat_implementasi' =>
                    $tingkatImplementasi,

                'cakupan_aktivitas' =>
                    $cakupanAktivitas,

                'durasi_kerjasama' =>
                    $durasiKerjasama,
            ]);

            c45ManajemenMitra::updateOrCreate(
                [
                    $table === 'mou'
                        ? 'mou_id'
                        : 'pks_id' => $mitraId,
                ],
                [
                    'mou_id' => $table === 'mou'
                        ? $mitraId
                        : null,

                    'pks_id' => $table === 'pks'
                        ? $mitraId
                        : null,

                    'nama_instansi' =>
                        $request->nama_instansi
```

```
                        ?? $mitra->nama_instansi
                        ?? '-',

                    'jumlah_mou' => $jumlahMou,
                    'jumlah_moa' => $jumlahMoa,
                    'jumlah_ia' => $jumlahIa,
                    'jumlah_kegiatan' => $jumlahKegiatan,

                    'jumlah_jenis_kegiatan' =>
                        $jumlahJenisKegiatan,

                    'kelengkapan_dokumen' =>
                        $kelengkapanDokumen,

                    'tingkat_implementasi' =>
                        $tingkatImplementasi,

                    'cakupan_aktivitas' =>
                        $cakupanAktivitas,

                    'durasi_kerjasama' =>
                        $durasiKerjasama,

                    'target_risiko' => $targetRisiko,
                ]
            );

            DB::commit();

            return back()->with(
                'success',
                'Analisis berhasil disimpan. Risiko: '
                . $targetRisiko
            );
        } catch (\Exception $e) {
            if (DB::transactionLevel() > 0) {
                DB::rollBack();
            }

            return back()
                ->with(
                    'error',
                    'Gagal menyimpan analisis: '
                    . $e->getMessage()
                )
                ->withInput();
        }
    }

    /**
     * Menentukan kategori kelengkapan dokumen C1.
     */
    private function hitungKelengkapanDokumen(
        $mou,
        $moa,
        $ia
    ): string {
        $jumlahJenisDokumen = 0;
```

```
        if ((int) $mou > 0) {
            $jumlahJenisDokumen++;
        }

        if ((int) $moa > 0) {
            $jumlahJenisDokumen++;
        }

        if ((int) $ia > 0) {
            $jumlahJenisDokumen++;
        }

        if ($jumlahJenisDokumen >= 3) {
            return 'Kuat';
        }

        if ($jumlahJenisDokumen === 2) {
            return 'Menengah';
        }

        return 'Lemah';
    }

    /**
     * Menentukan tingkat implementasi C2.
     */
    private function hitungTingkatImplementasi(
        $jumlahKegiatan
    ): string {
        if ((int) $jumlahKegiatan >= 4) {
            return 'Tinggi';
        }

        if ((int) $jumlahKegiatan >= 2) {
            return 'Sedang';
        }

        return 'Rendah';
    }

    /**
     * Menentukan cakupan aktivitas C3.
     */
    private function hitungCakupanAktivitas(
        $jumlahJenisKegiatan
    ): string {
        if ((int) $jumlahJenisKegiatan >= 2) {
            return 'Tinggi';
        }

        if ((int) $jumlahJenisKegiatan === 1) {
            return 'Sedang';
        }

        return 'Rendah';
    }

    /**
     * Menghitung durasi kerja sama dalam tahun.
```

```
     */
    private function hitungDurasiTahun(
        $tanggalAwal,
        $tanggalBerakhir
    ) {
        if (!$tanggalAwal || !$tanggalBerakhir) {
            return null;
        }

        try {
            $awal = new \DateTime($tanggalAwal);
            $akhir = new \DateTime($tanggalBerakhir);

            if ($akhir < $awal) {
                return null;
            }

            return round(
                $awal->diff($akhir)->days / 365,
                2
            );
        } catch (\Exception $e) {
            return null;
        }
    }

    /**
     * Menentukan kategori durasi kerja sama C4.
     */
    private function hitungDurasiKerjasama(
        $durasiTahun
    ): string {
        if ($durasiTahun === null) {
            return 'Pendek';
        }

        if ($durasiTahun >= 5) {
            return 'Panjang';
        }

        if ($durasiTahun >= 3) {
            return 'Sedang';
        }

        return 'Pendek';
    }

    /**
     * Menggunakan pohon keputusan jika tersedia.
     */
    private function predictOrTemporary(
        array $row
    ): string {
        $hasAll =
            !empty($row['kelengkapan_dokumen'])
            && !empty($row['tingkat_implementasi'])
            && !empty($row['cakupan_aktivitas'])
            && !empty($row['durasi_kerjasama']);
```

```
        $pohonTersedia = DB::table($this->tablePohon)
            ->where('is_root', 1)
            ->exists();

        if ($hasAll && $pohonTersedia) {
            return $this->predict($row);
        }

        return 'Sedang';
    }

    /**
     * Menelusuri pohon keputusan sampai leaf node.
     */
    private function predict(array $row): string
    {
        $root = DB::table($this->tablePohon)
            ->where('is_root', 1)
            ->first();

        if (!$root) {
            return 'Sedang';
        }

        $node = $root;
        $path = 'ROOT';

        while ($node && (int) $node->is_leaf !== 1) {
            $atribut = $node->atribut;
            $nilai = $row[$atribut] ?? null;

            if (!$atribut || !$nilai) {
                return $this
                    ->fallbackDecisionFromNode($node);
            }

            $path .= '|'
                . $atribut
                . '='
                . $nilai;

            $child = DB::table($this->tablePohon)
                ->where('parent', $path)
                ->orderBy('id_pohon', 'ASC')
                ->first();

            if (!$child) {
                return $this
                    ->fallbackDecisionFromNode($node);
            }

            $node = $child;
        }

        return $node->keputusan
            ?? $this->fallbackDecisionFromNode($node);
    }

    /**
```

```
     * Mengambil kelas mayoritas sebagai keputusan cadangan.
     */
    private function fallbackDecisionFromNode(
        $node
    ): string {
        if (!$node) {
            return 'Sedang';
        }

        $counts = [
            'Rendah' =>
                (int) ($node->jumlah_rendah ?? 0),

            'Sedang' =>
                (int) ($node->jumlah_sedang ?? 0),

            'Tinggi' =>
                (int) ($node->jumlah_tinggi ?? 0),
        ];

        arsort($counts);

        return array_key_first($counts)
            ?: 'Sedang';
    }
}
```