LAMPIRAN

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama : **Michael Kevin Adinata**
Nim : **2118069**
Jurusan : **Teknik Informatika S-1**
Judul : **Penerapan Algoritma K-Means untuk Klasterisasi Daerah Rawan Bencana Alam di Jawa Timur**

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

Hari : **Selasa**
Tanggal : **21 Januari 2025**
Nilai : **81 (A)**

Panitia Ujian Skripsi :
Ketua Majelis Penguji

Yosep Agus Pranoto ST., MT
NIP.P 1031000432

Anggota Penguji :

Dosen Penguji I

Joseph Dedy Irawan, ST., MT
NIP. 197404162005011002

Dosen Penguji II

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

Dipindai dengan CamScanner

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 SKRIPSI

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

NAMA : Michael Kevin Adinata
NIM : 2118069
JURUSAN : Teknik Informatika S-1
JUDUL : Penerapan Algoritma K-Means untuk Klasterisasi Daerah Rawan Bencana Alam di Jawa Timur

No.	Penguji	Tanggal	Uraian	Paraf
1.	Penguji I	21 Januari 2025	1. Menambahkan data mentah pada lampiran laporan. 2. Tujuan akhir harus jelas pada laporan. 3. Menambahkan evaluasi klasterisasi. 4. Mencoba mencari dan membandingkan hasil klasterisasi dengan data asli tahun 2023-2024	
2.	Penguji II	21 Januari 2025	1. Menambahkan hasil setiap iterasi pada laporan. 2. Perbaikan pada laporan. 3. Menambahkan evaluasi klasterisasi. 4. Kesimpulan	

Anggota Penguji :

Dosen Penguji I

Joseph Dedy Irawan, ST., MT
NIP. 197404162005011002

Dosen Penguji II

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

Mengetahui :

Dosen Pembimbing I

Ali Mahmudi, B Eng. PhD
NIP.Y 1031000429

Dosen Pembimbing II

Yosep Agus Pranoto ST., MT
NIP.P 1031000432

Dipindai dengan CamScanner

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

Malang, 04 Oktober 2024

Nomor : ITN-883/III.INF/TA/2024
Lampiran : ---
Perihal : *Pembimbing Utama Skripsi*

Kepada : **Yth. Bpk/Ibu Ali Mahmudi B.Eng., Ph.D.**
Dosen Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

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

Nama : Michael Kevin Adinata
Nim : 2118069
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 Agustus 2024 s/d 21 Pebruari 2025

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

Dipindai dengan CamScanner

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

Malang, 04 Oktober 2024

Nomor : ITN-883/III.INF/TA/2024
Lampiran : —
Perihal : *Pembimbing Pendamping Skripsi*

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

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

Nama : Michael Kevin Adinata
Nim : 2118069
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 Agustus 2024 s/d 21 Pebruari 2025

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
K e t u a,

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

Form S-4a

Dipindai dengan CamScanner

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

FORMULIR BIMBINGAN SKRIPSI

Nama : michael kevin adinata
Nim : 2118069
Masa Bimbingan : 21 Agustus 2024 - 21 Februari 2025
Judul Skripsi : Penerapan algoritma K-Means untuk klasterisasi daerah rawan bencana alam di Jawa Timur

No.	Tanggal	Uraian	Paraf Pembimbing
1	15/10 24	Saran: salah satu wilayah kota / kab. di Jatim.	
2	22/10 24	Demo front end. Kota/Kab. Kec. Kriteria. Jenis. Data. Proses	
3	28/10 24	Demo web.	
4	21/11 24	Demo. Proses perbaikan.	
5	26/11 24	Draf makalah. Tunjukkan proses di web.	
6	28/11 24	Demo proses + tunjukkan iterasi	
7	29/11 24	Draf. skripsi	
8	15/01 25	konsultasi seminar hasil, pengujian user disesuaikan (kuisioner)	
9	16/01 25	revisi laporan skripsi	
10	17/01 25	finalisasi laporan & ACC kompre	

Malang, ...
Dosen Pembimbing

(Ali Mahmudi B.Eng,PhD)
NIP. P. 1031000429

CS Dipindai dengan CamScanner

FORMULIR BIMBINGAN SKRIPSI

Nama : Michael Kevin Adinata

Nim : 2118069

Masa Bimbingan : 21 Agustus 2024 - 21 Februari 2025

Judul Skripsi : Penerapan algoritma K-Means untuk klasterisasi daerah rawan bencana alam di Jawa Timur

No.	Tanggal	Uraian	Paraf Pembimbing
1.	17 Oktober 2024	Klasterisasi dijadikan satu tidak per bencana (untuk perhitungan), lalu detail tetap masing-masing, klasterisasi masuk logger, tambah DFD	
2.	25 Oktober 2024	konsultasi progress dari pengerjaan website, konsultasi proses klasterisasi	
3.	31 oktober 2024	Demo website, ppt ditambahkan seperti semero, logger ditambahkan jarak ke centroid, menambahkan peta hasil klasterisasi	
4.	18 November 2024	konsultasi revisi semprog	
5.	20 November 2024	Demo web, pemetaan.	
6.	28 November 2024	User ada 2 role (Superadmin, admin), laporan (pengujian user), jurnal	
7.	06 Desember 2024	laporan pengujian user	
8.	15 Jan 2025	konsultasi seminar hasil, pengujian user disesuaikan	
9.	16 Jan 2025	menyesuaikan hasil revisi laporan (pengujian user)	
10.	17 Jan 2025	finishing laporan & ACC kompre	

Malang, 17/1/2025

Dosen Pembimbing

Yosep Agus Pranoto, ST., MT.

NIP.P 1031000432

CS Dipindai dengan CamScanner

PEMERINTAH PROVINSI JAWA TIMUR
BADAN PENANGGULANGAN BENCANA DAERAH
Jalan Letjen S. Parman Nomor 55, Waru, Waru, Sidoarjo, Jawa Timur 61256
Telepon (031) 8550222/Faksimile (031) 8550101
Laman bpbd.jatimprov.go.id, Pos-el mail@bpbd.jatimprov.go.id

SURAT PERNYATAAN

Yang bertandatangan di bawah ini :

Nama	: Andhika Nurrahmad Sudigda
Jabatan	: Sekretaris
Instansi/Lembaga	: BPBD Jawa Timur
No. Telepon/HP	: 082261008645
Alamat	: Jl. Letjend. S. Parman No.55, Krajan Kulon, Waru, Kec. Waru, Kabupaten Sidoarjo, Jawa Timur 61256

Dengan ini menyatakan bersedia menjadi mitra dalam pembuatan skripsi dari

Nama	: Michael Kevin Adinata
NIM	: 2118069
Judul Skripsi	: Penerapan Algoritma K-Means untuk Klasterisasi Daerah Rawan Bencana Alam di Jawa Timur

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 : Data daerah bencana alam tahun 2021 - 2022

Sidoarjo, 06 Januari 2025
Yang membuat pernyataan

METERAI TEMPEL
096AMX108924447

(Andhika Nurrahmad Sudigda)

Dipindai dengan CamScanner

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	: Michael Kevin Adinata
NIM	: 21180.69
Fakultas / Jurusan	: Teknologi Industri / Teknik Informatika S-1
Email	: kevinadinata08@gmail.com
No. Tlp	: 085790291176
Judul/ Jml artikel	: Penerapan algoritma K-Means untuk klasterasasi Daerah Rawan Bencana Alam di Jawa Timur

Karya ilmiah yang bersangkutan di atas melalui proses cek plagiatsi menggunakan aplikasi trunitin dengan hasil kemiripan (Similarty) Sebesar.....1.1..... %
Demikian surat keterangan ini dibuat agar dapat dipergunakan sebagaimana mestinya.

Malang, 11 Februari 2025

Mahasiswa

michael kevin adinata

Pelaksana,

SKRIPSI PENERAPAN ALGORITMA K-MEANS UNTUK KLASTERISASI DAERAH RAWAN BENCANA ALAM DI JAWA TIMUR

ORIGINALITY REPORT

11%	0%	10%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

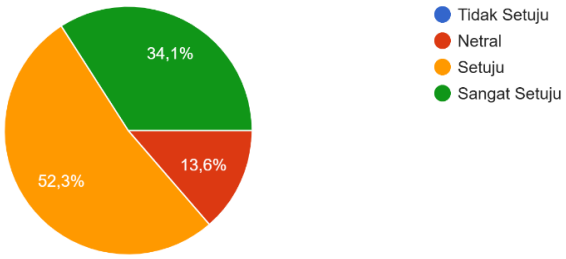
PRIMARY SOURCES

1	Michael Kevin Adinata, Ali Mahmudi, Yosep Agus Pranoto. "Klasterisasi Daerah Rawan Bencana Alam Menggunakan Algoritma K-Means", Infotek: Jurnal Informatika dan Teknologi, 2025 Publication	8%
2	Submitted to Xavier University Student Paper	2%
3	Nayla Salsabila, Karina Aulisari, Hani Zulfia Zahro. "Penerapan Algoritma K-Means Untuk Klasterisasi Produktivitas Tanaman Jahe", Infotek: Jurnal Informatika dan Teknologi, 2025 Publication	2%

Exclude quotes	Off	Exclude matches	< 2%
Exclude bibliography	On		

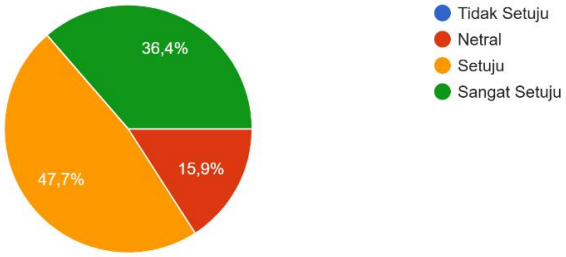
Apakah data kabupaten kota, data kecamatan, dan data bencana alam yang ditampilkan sudah informatif?

44 jawaban



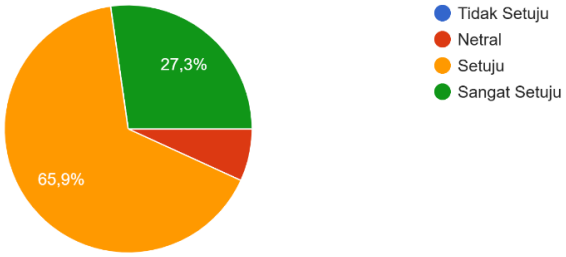
Apakah tampilan UI (User Interface) aplikasi menarik untuk digunakan?

44 jawaban



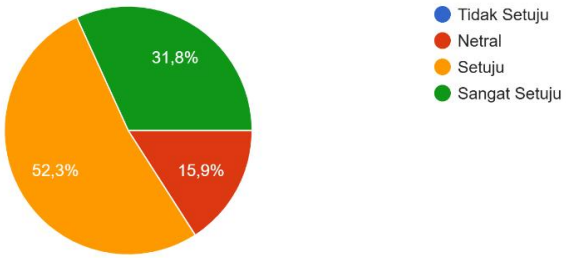
Apakah menu halaman kabupaten kota, halaman kecamatan, halaman bencana alam mudah untuk dipahami dan dioperasikan?

44 jawaban



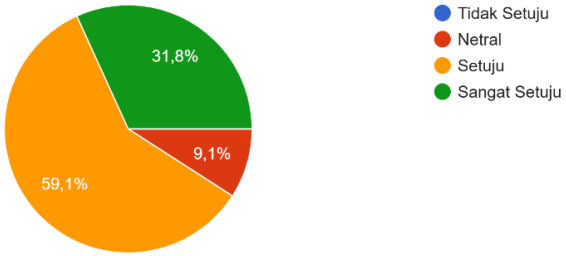
Apakah tampilan data hasil klasterisasi yang ditampilkan sudah informatif?

44 jawaban



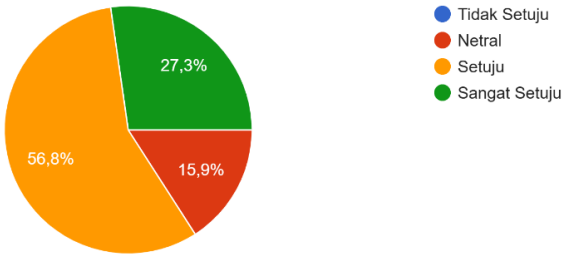
Apakah tampilan peta hasil clustering yang ditampilkan sudah informatif?

44 jawaban



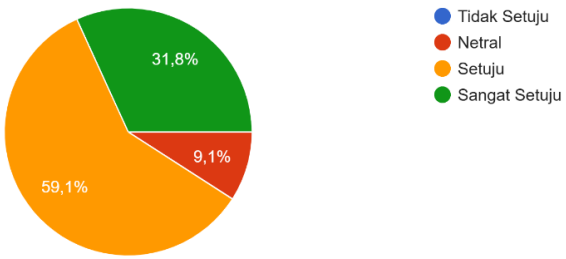
Apakah hasil dari klasterisasi dan pemetaan di aplikasi sudah sesuai dengan yang diharapkan?

44 jawaban



Apakah aplikasi ini memberikan kemudahan dalam melakukan pengelompokan data bencana alam di Jawa Timur?

44 jawaban



LAMPIRAN DATA

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021	Kabupaten Bangkalan	Socah	Angin Kencang	2	33	0
2021		Bangkalan	Angin Puting Beliung	1	52	0
2021		Blega	Banjir	1	2399	0
2021		Modung	Angin Kencang	1	31	0
2021		Kwanyar	Angin Kencang	1	5	0
2021	Kabupaten Banyuwangi	Gambiran	Angin Kencang	1	14	0
2021			Angin Puting Beliung	1	36	1
2021		Glenmore	Angin Kencang	1	1	0
2021		Kabat	Angin Kencang	1	13	0
2021		Cluring	Angin Puting Beliung	1	29	1
2021		Banyuwangi	Banjir	3	1094	0
2021		Wongsorejo	Banjir	2	300	0
2021		Muncar	Banjir	1	500	0
2021		Srono	Angin Kencang	1	1	0
2021		Genteng	Angin Puting Beliung	1	16	0
2021		Licin	Tanah Longsor	1	3	6
2021	Kota Batu	Bumiaji	Banjir	2	55	3
2021			Tanah Longsor	3	0	0
2021		Batu	Gerakan Tanah	1	3	0
2021			Tanah Longsor	2	18	0
2021			Banjir	1	53	4
2021	Kabupaten Blitar	Wonodadi	Banjir	1	65	0
2021			Angin Kencang	1	20	0
2021		Srengat	Angin Kencang	1	4	0
2021		Sanankulon	Angin Kencang	1	3	0

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021		Udanawu	Angin Kencang	1	4	0
2021		Ponggok	Angin Kencang	1	14	5
2021		Wonodadi	Angin Kencang	1	24	0
2021	Kota Blitar	Kepanjenkidul	Angin Kencang	1	3	0
2021		Sanawetan	Angin Kencang	1	3	0
2021	Bojonegoro Kabupaten	Ngraho	Angin Kencang	1	5	0
2021		Kanor	Banjir	1	103	0
2021		Temayang	Banjir	1	120	0
2021		Gondang	Banjir	3	565	0
2021		Sukosewu	Banjir	2	87	0
2021		Sekar	Banjir	4	431	1
2021		Dander	Banjir	1	229	0
2021		Trucuk	Banjir	1	71	0
2021		Malo	Banjir	1	133	0
2021		Bojonegoro	Banjir	2	507	0
2021	Kabupaten Bondowoso	Wonosari	Angin Kencang	1	33	0
2021		Tamanan	Angin Kencang	2	22	0
2021		Binakal	Angin Kencang	1	23	0
2021		Tenggarang	Angin Kencang	1	23	0
2021		Wringin	Angin Kencang	1	27	0
2021		Botolinggo	Angin Kencang	1	27	0
2021		Sempol	Banjir Bandang	1	6	0
2021			Tanah Longsor	1	1	0
2021		Ijen	Tanah Longsor	1	1	0
2021	Kabupaten Gresik	Cerme	Angin Kencang	1	20	0
2021			Banjir	4	455	0
2021		Benjeng	Angin Kencang	1	11	0
2021			Banjir	8	1087	0
2021		Balongpanggang	Banjir	9	1193	0
2021		Bungah	Banjir	1	3	0

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021		Ujungpangkah	Banjir	1	0	0
2021		Kedamean	Banjir	3	646	0
2021		Menganti	Banjir	3	624	0
2021		Driyorejo	Banjir	1	119	0
2021		Duduksampeyan	Banjir	2	308	0
2021		Sangkapura	Banjir	1	4	0
2021	Kabupaten Jember	Sukowono	Angin Kencang	1	1	0
2021		Ledokombo	Angin Kencang	1	9	0
2021		Ambulu	Angin Kencang	1	14	0
2021			Banjir	2	54	0
2021		Kalisat	Angin Kencang	1	13	0
2021			Banjir	1	16	0
2021		Arjasa	Angin Kencang	1	5	0
2021		Balung	Angin Kencang	1	5	0
2021		Jenggawah	Angin Kencang	1	5	0
2021		Patrang	Banjir	2	112	0
2021			Angin Kencang	1	5	0
2021		Pakusari	Angin Kencang	1	22	0
2021			Banjir	1	5	0
2021		Umbulsari	Banjir	1	296	0
2021			Angin Kencang	1	0	0
2021		Kaliwates	Banjir	2	361	0
2021		Bangsalsari	Banjir	3	31	0
2021		Rambipuji	Banjir	2	54	0
2021		Silo	Banjir	1	3	0
2021		Jombang	Banjir	1	100	0
2021		Puger	Banjir	2	560	0
2021			Angin Kencang	1	267	1
2021		Gumukmas	Banjir	1	35	0
2021			Angin Kencang	1	57	1
2021		Tanggul	Banjir	1	100	0

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021			Tanah Longsor	1	76	0
2021		Semboro	Banjir	1	869	0
2021		Wuluhan	Banjir	1	59	0
2021		Tempurejo	Banjir	2	976	0
2021		Jelbuk	Banjir	1	12	0
2021		Sumbersari	Banjir	1	28	0
2021		Sumberbaru	Tanah Longsor	1	356	0
2021	Kabupaten Jombang	Diwek	Angin Kencang	1	24	0
2021		Kesamben	Banjir	2	510	0
2021			Angin Kencang	2	28	0
2021		Bandar Kedungmulyo	Banjir	1	2067	0
2021			Angin Kencang	1	33	0
2021		Mojoagung	Banjir	9	3765	0
2021		Mojowarno	Banjir	4	1285	0
2021		Peterongan	Banjir	1	526	0
2021		Wonosalam	Banjir	1	3	0
2021		Bareng	Banjir	1	44	0
2021	Kabupaten Kediri	Purwoasri	Angin Kencang	1	21	0
2021		Gurah	Banjir	1	73	0
2021		Gampengrejo	Banjir	1	422	0
2021		Ngasem	Banjir	2	301	0
2021	Kota Kediri	Mojoroto	Banjir	1	7	0
2021	Kabupaten Lamongan	Glagah	Banjir	2	461	0
2021			Angin Kencang	1	230	0
2021		Pucuk	Banjir	1	198	0
2021		Babat	Banjir	2	2850	0
2021		Kalitengah	Banjir	3	514	0
2021		Turi	Banjir	2	1313	0
2021		Deket	Banjir	2	158	0
2021		Karangbinangun	Banjir	2	69	0
2021		Tikung	Banjir	1	6	0
2021		Lamongan	Banjir	2	4	0

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021			Gerakan Tanah	1	0	0
2021		Modo	Banjir	1	25	0
2021		Sugio	Banjir	1	45	0
2021		Laren	Tanah Longsor	1	2	0
2021	Kabupaten Lumajang	Candipuro	Angin Kencang	1	17	30
2021			Banjir	1	0	0
2021			Gunung Api	1	578	0
2021		Pasrujambe	Angin Puting Beliung	2	108	0
2021		Kunir	Banjir	2	40	0
2021		Kedungjajang	Banjir	1	20	0
2021			Tanah Longsor	1	8	0
2021		Sukodono	Banjir	1	1273	0
2021		Tekung	Banjir	1	48	0
2021		Lumajang	Banjir	2	20	0
2021		Rowokangkung	Banjir	1	97	0
2021		Pronojiwo	Gunung Api	1	577	31
2021	Kabupaten Madiun	Saradan	Angin Kencang	1	8	0
2021			Banjir	1	34	0
2021			Tanah Longsor	1	0	0
2021		Kare	Tanah Longsor	3	2	0
2021			Angin Kencang	1	8	0
2021		Wungu	Angin Kencang	1	8	0
2021			Banjir	2	30	0
2021		Wonoasri	Angin Puting Beliung	1	261	3
2021			Banjir	2	186	0
2021		Sawahan	Angin Puting Beliung	1	1	0
2021		Mejayan	Banjir	2	116	0
2021		Geger	Banjir	1	0	0
2021		Gemarang	Banjir	2	20	0

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021			Tanah Longsor	1	1	0
2021		Madiun	Banjir	1	150	0
2021		Balerejo	Banjir	2	389	0
2021		Dagangan	Tanah Longsor	1	3	0
2021	Kota Madiun	Manguharjo	Banjir	1	1	0
2021	Kabupaten Magetan	Kartoharjo	Banjir	1	21	0
2021		Barat	Banjir	1	92	0
2021		Ngariboyo	Banjir	1	42	0
2021			Angin Kencang	1	6	0
2021		Kawedanan	Banjir	1	21	0
2021		Nguntoronadi	Banjir	1	3	0
2021		Karas	Tanah Longsor	1	1	0
2021		Plaosan	Tanah Longsor	1	4	0
2021	Kabupaten Malang	Singosari	Angin Kencang	1	13	0
2021		Ngantang	Angin Kencang	1	4	0
2021			Gerakan Tanah	2	31	0
2021			Tanah Longsor	2	1	0
2021		Pakis	Angin Kencang	1	39	0
2021		Pakisaji	Angin Kencang	1	82	0
2021		Tirtoyuo	Tanah Longsor	1	9	0
2021		Pujon	Tanah Longsor	1	1	0
2021		Ampelgading	Tanah Longsor	1	18	0
2021		Wajak	Tanah Longsor	1	5	0
2021		Wonosari	Tanah Longsor	1	0	1
2021	Kota Malang	Klojen	Angin Kencang	2	6	1
2021			Banjir	2	221	0
2021			Tanah Longsor	1	12	1
2021		Sukun	Angin Kencang	1	218	1

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021			Banjir	1	141	0
2021		Blimbing	Angin Kencang	1	15	0
2021			Banjir	1	43	0
2021		Lowokwaru	Banjir	1	61	0
2021		Kedungkandang	Angin Kencang	1	0	0
2021			Banjir	1	5	0
2021			Tanah Longsor	1	1	0
2021	Kabupaten Mojokerto	Trowulan	Angin Kencang	1	58	0
2021		Bangsal	Angin Kencang	1	7	0
2021		Jetis	Angin Kencang	2	51	2
2021		Mojoanyar	Angin Kencang	1	17	0
2021		Mojosari	Angin Kencang	1	30	0
2021			Banjir	2	60	0
2021		Kemlagi	Banjir	1	172	0
2021		Dawarblandong	Banjir	1	33	0
2021		Pungging	Banjir	1	86	0
2021		Sooko	Banjir	2	762	0
2021		Pacet	Tanah Longsor	1	1	0
2021	Kabupaten Nganjuk	Pace	Banjir	3	15	0
2021		Nganjuk	Banjir	3	132	0
2021		Prambon	Banjir	2	891	0
2021		Loceret	Banjir	3	26	0
2021		Berbek	Banjir	2	145	0
2021		Sawahan	Tanah Longsor	1	1	0
2021		Ngetos	Tanah Longsor	1	64	37
2021	Kabupaten Ngawi	Pangkur	Banjir	1	8	0
2021		Kwadungan	Banjir	1	6	0
2021	Kabupaten Pamekasan	Pademawu	Angin Kencang	1	77	0
2021		Palengaan	Banjir	1	1	0
2021		Pamekasan	Banjir	2	976	0
2021		Pasean	Tanah Longsor	1	1	6

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021	Kabupaten Pasuruan	Beji	Angin Kencang	1	27	0
2021			Banjir	3	1195	0
2021		Pandaan	Angin Kencang	1	0	0
2021		Gempol	Banjir	2	1771	2
2021			Angin Kencang	1	0	0
2021		Bangil	Banjir	7	1889	0
2021			Angin Kencang	2	110	0
2021		Pohjentrek	Angin Kencang	1	15	0
2021		Rejoso	Banjir	3	565	0
2021			Angin Kencang	1	3	0
2021		Pasrepan	Angin Kencang	1	0	0
2021		Lumbang	Angin Puting Beliung	1	14	0
2021		Kraton	Banjir	2	776	0
2021		Grati	Banjir	3	990	0
2021		Winongan	Banjir	3	947	0
2021		Gondang Wetan	Banjir	2	700	0
2021		Tosari	Tanah Longsor	1	5	0
2021	Kota Pasuruan	Purworejo	Angin Kencang	1	6	0
2021			Banjir	3	335	0
2021		Panggungrejo	Angin Kencang	1	6	0
2021			Banjir	8	1281	0
2021		Gadingrejo	Angin Kencang	1	7	0
2021			Banjir	4	255	0
2021		Bugul Kidul	Banjir	7	2728	0
2021	Kabupaten Ponorogo	Sampung	Banjir	1	0	0
2021		Kauman	Banjir	1	18	0
2021		Ponorogo	Banjir	1	1	0
2021		Pulung	Gerakan Tanah	1	10	0
2021	Kabupaten Probolinggo	Sumber	Angin Kencang	1	11	0
2021		Bantaran	Angin Kencang	1	7	0

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021		Dringu	Banjir	4	4828	0
2021		Leces	Banjir	2	1132	0
2021		Gending	Banjir	1	125	0
2021		Paiton	Banjir	1	150	0
2021	Kota Probolinggo	Wonoasih	Banjir	2	196	0
2021		Kedopok	Banjir	1	30	0
2021		Mayangan	Banjir	1	107	0
2021		Kanigaran	Banjir	1	108	0
2021	Kabupaten Sampang	Sokobanah	Angin Puting Beliung	1	13	0
2021		Sampang	Angin Puting Beliung	1	32	1
2021			Banjir	1	13	0
2021		Camplong	Angin Puting Beliung	1	17	0
2021		Jrengik	Banjir	1	365	0
2021	Kabupaten Sidoarjo	Waru	Angin Kencang	2	59	0
2021		Taman	Angin Kencang	1	2	0
2021		Gedangan	Angin Kencang	2	71	0
2021		Sedati	Angin Kencang	1	3	0
2021			Angin Puting Beliung	1	34	0
2021		Tulangan	Angin Kencang	2	6	0
2021		Jabon	Angin Kencang	1	10	0
2021		Sidoarjo	Angin Kencang	2	19	0
2021			Banjir	1	1233	0
2021		Tanggulangin	Angin Kencang	1	16	0
2021			Banjir	5	2198	0
2021		Buduran	Angin Kencang	1	36	0
2021		Wonoayu	Angin Kencang	3	76	0
2021		Candi	Angin Kencang	1	16	0

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021			Angin Puting Beliung	1	23	0
2021			Banjir	2	1354	0
2021		Krembung	Angin Kencang	1	1	0
2021		Tarik	Angin Kencang	1	13	0
2021		Porong	Angin Kencang	1	157	0
2021	Kabupaten Situbondo	Besuki	Abrasi	1	4	0
2021			Banjir	2	179	0
2021			Angin Kencang	1	22	0
2021		Mangaran	Angin Kencang	1	112	0
2021		Panji	Angin Kencang	1	28	0
2021		Banyuputih	Banjir	2	128	0
2021		Jangkar	Banjir	1	112	0
2021		Situbondo	Tanah Longsor	1	1	0
2021	Kabupaten Sumenep	Saronggi	Angin Kencang	1	17	1
2021		Kalianget	Angin Kencang	1	35	0
2021			Angin Puting Beliung	1	22	3
2021		Ganding	Banjir	1	1	0
2021		Lenteng	Banjir	1	60	0
2021	Kabupaten Tuban	Kerek	Angin Kencang	2	11	0
2021		Widang	Angin Kencang	1	12	0
2021			Banjir	1	251	0
2021		Tambakboyo	Angin Puting Beliung	1	35	0
2021		Plumpang	Banjir	3	715	0
2021		Merakurak	Banjir	2	160	0
2021		Semanding	Banjir	1	12	0
2021		Parengan	Banjir	1	2060	0
2021		Jenu	Kebakaran	1	1	0
2021		Rejotangan	Angin Kencang	2	44	0

Thn	Kab/Kota	Kecamatan	Jenis Bencana	Frekuensi Kejadian	Total Kerusakan	Total Korban Jiwa
2021	Kabupaten Tulungagung	Ngunut	Angin Kencang	1	66	0
2021		Pagerwojo	Tanah Longsor	2	42	0
2021		Sendang	Tanah Longsor	1	23	0

Untuk data bencana alam selengkapnya bisa di cek pada QR Code dibawah ini:

LAMPIRAN SOURCE CODE

1. Source Code KlasterisasiController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\TbClustering;
use App\Models\TbDataBencana;
use App\Models\TbSilhouetteScore;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use RealRashid\SweetAlert\Facades\Alert;
class KlasterisasiController extends Controller
{
    public function index()
    {
        $tahunList = TbDataBencana::selectRaw('tahun')
            ->distinct()
            ->orderBy('tahun', 'asc')
            ->pluck('tahun');
        return view('klasterisasi.index',
compact('tahunList'));
    }
    public function fetchData(Request $request)
    {
        $tahunDipilih = $request->input('tahun');
        if (!$tahunDipilih) {
            return response()->json(['status' => 'error',
'message' => 'Tahun belum dipilih.'], 400);
        }
        $data = TbClustering::with('tb_kecamatan',
'tb_kotakab')->where('tahun', $tahunDipilih)->get();
        if ($data->isEmpty()) {
            return response()->json(['status' => 'empty',
'message' => 'Data klasterisasi tidak ditemukan untuk tahun
ini.']);
        }
        return response()->json(['status' => 'success',
'data' => $data]);
    }
    public function show($id)
    {
        $clustering = TbClustering::selectRaw('
                tb_clustering.cluster,
                tb_kotakab.nama_kotakab,
                tb_kecamatan.nama_kecamatan,
                tb_jenisbencana.nama_bencana,
                tb_databencana.tahun,
                tb_databencana.frekuensi_kejadian,
                tb_databencana.total_kerusakan,
                tb_databencana.total_korban
            ')
            ->join('tb_kecamatan',
'tb_clustering.id_kecamatan', '=', 'tb_kecamatan.id')
            ->join('tb_kotakab', 'tb_kecamatan.id_kotakab',
'=', 'tb_kotakab.id')
            ->join('tb_databencana',
'tb_databencana.id_kecamatan', '=', 'tb_kecamatan.id')
```

```
            ->join('tb_jenisbencana', 'tb_jenisbencana.id',
'=', 'tb_databencana.id_jenisbencana')
            ->where('tb_kecamatan.id', '=', $id)
            ->get();
        if ($clustering->isEmpty()) {
            Alert::error('Data not found', 'Data tidak
ditemukan');
            return redirect('klasterisasi/hasil');
        }
        return view('klasterisasi.detail',
compact('clustering'));
    }
    public function prosesKlasterisasi(Request $request)
    {
        $tahun = $request->input('tahun');
        $dataBencana = TbDatabencana::selectRaw('
                tb_kotakab.id as id_kotakab,
                tb_kotakab.nama_kotakab as nama_kabupaten,
                tb_kecamatan.id as id_kecamatan,
                tb_kecamatan.nama_kecamatan,
                tb_databencana.tahun,
                SUM(tb_databencana.frekuensi_kejadian) as
total_frekuensi,
                SUM(tb_databencana.total_kerusakan) as
total_kerusakan,
                SUM(tb_databencana.total_korban) as
total_korban
            ')
            ->join('tb_kecamatan',
'tb_databencana.id_kecamatan', '=', 'tb_kecamatan.id')
            ->join('tb_kotakab', 'tb_kecamatan.id_kotakab',
'=', 'tb_kotakab.id')
            ->where('tb_databencana.tahun', $tahun)
            ->groupBy('tb_kecamatan.nama_kecamatan',
'tb_kotakab.nama_kotakab', 'tb_kotakab.id',
'tb_kecamatan.id', 'tb_databencana.tahun')
            ->orderBy('tb_kotakab.nama_kotakab')
            ->orderBy('tb_kecamatan.nama_kecamatan')
            ->get();
        $data = [];
        foreach ($dataBencana as $item) {
            $data[] = [
                'id_kotakab' => $item->id_kotakab,
                'nama_kabupaten' => $item->nama_kabupaten,
                'id_kecamatan' => $item->id_kecamatan,
                'nama_kecamatan' => $item->nama_kecamatan,
                'total_frekuensi' => $item->total_frekuensi,
                'total_kerusakan' => $item->total_kerusakan,
                'total_korban' => $item->total_korban,
                'tahun' => $item->tahun,
            ];
        }
        $currentHash = md5(json_encode($data));
        $lastHash = TbClustering::where('tahun', $tahun)-
>orderBy('created_at', 'desc')->value('data_hash');

        if ($currentHash === $lastHash) {
            Alert::error('Peringatan', 'Tidak ada perubahan
data bencana, proses klasterisasi tidak dilakukan.');
```

```
            return redirect('klasterisasi/hasil');
        }
        $jumlahCluster = 3;
        $clusters = $this->kMeans($data, $jumlahCluster,
$tahun);
        TbClustering::where('tahun', $tahun)->delete();
        $this->insertHasilKlasterisasi($clusters,
$currentHash);

        Alert::success('Success', 'Klasterisasi Berhasil');
        return redirect('klasterisasi/hasil');
    }
    private function kMeans($data, $k, $tahun)
    {
        $centroids = $this->initCentroids($data, $k);
        $iterations = 100;
        for ($i = 0; $i < $iterations; $i++) {
            $clusters = $this->assignClusters($data,
$centroids);
            $newCentroids = $this->updateCentroids($data,
$clusters['clusters'], $k);
            $this->insertIterationData($i + 1, $this-
>labelCentroids($centroids), $clusters['clusters'],
$clusters['euclidean_distances'], $tahun);
            if ($centroids == $newCentroids) {
                break;
            }
            $centroids = $newCentroids;
        }
        ksort($clusters['clusters']);
        $averageSilhouetteScore = $this-
>calculateSilhouetteScore($data, $clusters['clusters'],
$centroids);
        $silhouetteScore = TbSilhouetteScore::where('tahun',
'=', $tahun)->first();
        if (!$silhouetteScore) {
            DB::table('tb_silhouette_score')->insert([
                'avg_silhouette_score' =>
$averageSilhouetteScore,
                'tahun' => $tahun,
            ]);
        }
        return $clusters['clusters'];
    }
    private function labelCentroids($centroids)
    {
        $labeledCentroids = [];
        foreach ($centroids as $index => $centroid) {
            $label = '';
            switch ($index) {
                case 0:
                    $label = 'C1';
                    break;
                case 1:
                    $label = 'C2';
                    break;
                case 2:
                    $label = 'C3';
                    break;
```

```
            }
            $labeledCentroids[] = [
                'label' => $label,
                'centroid' => $centroid
            ];
        }
        return $labeledCentroids;
    }
    private function insertHasilKlasterisasi($clusters,
$currentHash)
    {
        foreach ($clusters as $index => $clusterData) {
            $clusterLabel = '';
            switch ($index) {
                case 0:
                    $clusterLabel = 'C1';
                    break;
                case 1:
                    $clusterLabel = 'C2';
                    break;
                case 2:
                    $clusterLabel = 'C3';
                    break;
            }
            foreach ($clusterData as $data) {
                DB::table('tb_clustering')->insert([
                    'id_kotakab' => $data['id_kotakab'],
                    'id_kecamatan' => $data['id_kecamatan'],
                    'frekuensi_kejadian' =>
$data['total_frekuensi'],
                    'total_kerusakan' =>
$data['total_kerusakan'],
                    'total_korban' => $data['total_korban'],
                    'cluster' => $clusterLabel,
                    'tahun' => $data['tahun'],
                    'created_at' => now(),
                    'data_hash' => $currentHash,
                ]);
            }
        }
    }
    private function initCentroids($data)
    {
        usort($data, function ($a, $b) {
            $sumA = $a['total_frekuensi'] +
$a['total_kerusakan'] + $a['total_korban'];
            $sumB = $b['total_frekuensi'] +
$b['total_kerusakan'] + $b['total_korban'];
            return $sumA <=> $sumB;
        });
        $centroids = [];
        $centroids[] = [
            'nama_kotakab' => $data[0]['nama_kabupaten'],
            'nama_kecamatan' => $data[0]['nama_kecamatan'],
            'total_frekuensi' => $data[0]['total_frekuensi'],
            'total_kerusakan' => $data[0]['total_kerusakan'],
            'total_korban' => $data[0]['total_korban'],
        ];
        $medianIndex = floor(count($data) / 2);
```

```
        $centroids[] = [
            'nama_kotakab' =>
$data[$medianIndex]['nama_kabupaten'],
            'nama_kecamatan' =>
$data[$medianIndex]['nama_kecamatan'],
            'total_frekuensi' =>
$data[$medianIndex]['total_frekuensi'],
            'total_kerusakan' =>
$data[$medianIndex]['total_kerusakan'],
            'total_korban' =>
$data[$medianIndex]['total_korban'],
        ];
        $lastIndex = count($data) - 1;
        $centroids[] = [
            'nama_kotakab' =>
$data[$lastIndex]['nama_kabupaten'],
            'nama_kecamatan' =>
$data[$lastIndex]['nama_kecamatan'],
            'total_frekuensi' =>
$data[$lastIndex]['total_frekuensi'],
            'total_kerusakan' =>
$data[$lastIndex]['total_kerusakan'],
            'total_korban' =>
$data[$lastIndex]['total_korban'],
        ];
        return $centroids;
    }
    private function assignClusters($data, $centroids)
    {
        $clusters = [];
        foreach ($data as $point) {
            $minDistance = PHP_INT_MAX;
            $closestCentroid = null;
            foreach ($centroids as $index => $centroid) {
                $distance =
sqrt(pow($point['total_frekuensi'] -
$centroid['total_frekuensi'], 2) +
pow($point['total_kerusakan'] - $centroid['total_kerusakan'],
2) + pow($point['total_korban'] - $centroid['total_korban'],
2));
                if ($distance < $minDistance) {
                    $minDistance = $distance;
                    $closestCentroid = $index;
                }
            }
            $clusters[$closestCentroid][] = $point;
        }
        $datas = [
            'clusters' => $clusters,
            'euclidean_distances' => array_map(function
($point) use ($centroids) {
                return array_map(function ($centroid) use
($point) {
                    return sqrt(
                        pow($point['total_frekuensi'] -
$centroid['total_frekuensi'], 2) +
                            pow($point['total_kerusakan'] -
$centroid['total_kerusakan'], 2) +
```

```
                            pow($point['total_korban'] -
$centroid['total_korban'], 2)
                    );
                }, $centroids);
            }, $data)
        ];
        return $datas;
    }
    private function calculateSilhouetteScore($data,
$clusters, $centroids)
    {
        $calculateEuclideanDistance = function ($point1,
$point2) {
            return sqrt(pow($point1['total_frekuensi'] -
$point2['total_frekuensi'], 2) +
                pow($point1['total_kerusakan'] -
$point2['total_kerusakan'], 2) +
                pow($point1['total_korban'] -
$point2['total_korban'], 2));
        };
        $silhouetteScores = [];
        $logDetails = [];
        foreach ($data as $index => $point) {
            $clusterId = $this->getClosestCluster($point,
$centroids);
            $sameClusterPoints = $clusters[$clusterId];
            $logDetails[$index] = [
                'point' => $point,
                'clusterId' => $clusterId,
                'a' => 0,
                'b' => PHP_INT_MAX,
                'silhouetteScore' => null,
                'intraClusterDistances' => [],
                'interClusterDistances' => [],
            ];
            $a = 0;
            foreach ($sameClusterPoints as $sameIndex =>
$samePoint) {
                if ($samePoint !== $point) {
                    $distance =
$calculateEuclideanDistance($point, $samePoint);

$logDetails[$index]['intraClusterDistances'][$sameIndex] =
$distance;
                    $a += $distance;
                }
            }
            $a /= count($sameClusterPoints) - 1;
            $logDetails[$index]['a'] = $a;

            $b = PHP_INT_MAX;
            foreach ($centroids as $centroidIndex =>
$centroid) {
                if ($centroidIndex !== $clusterId) {
                    $otherClusterPoints =
$clusters[$centroidIndex];
                    $bClusterDistance = 0;
                    foreach ($otherClusterPoints as
$otherIndex => $otherPoint) {
```

```
                        $distance =
$calculateEuclideanDistance($point, $otherPoint);

$logDetails[$index]['interClusterDistances'][$otherIndex] =
$distance;
                        $bClusterDistance += $distance;
                    }
                    $bClusterDistance /=
count($otherClusterPoints);
                    if ($bClusterDistance < $b) {
                        $b = $bClusterDistance;
                    }
                }
            }
            $logDetails[$index]['b'] = $b;
            $silhouetteScore = ($b - $a) / max($a, $b);
            $logDetails[$index]['silhouetteScore'] =
$silhouetteScore;
            $silhouetteScores[] = $silhouetteScore;
        }
        $averageSilhouetteScore =
array_sum($silhouetteScores) / count($silhouetteScores);
        $logDetails['averageSilhouetteScore'] =
$averageSilhouetteScore;

        return $averageSilhouetteScore;
    }
    private function getClosestCluster($point, $centroids)
    {
        $minDistance = PHP_INT_MAX;
        $closestClusterId = null;
        foreach ($centroids as $index => $centroid) {
            $distance = sqrt(pow($point['total_frekuensi'] -
$centroid['total_frekuensi'], 2) +
                pow($point['total_kerusakan'] -
$centroid['total_kerusakan'], 2) +
                pow($point['total_korban'] -
$centroid['total_korban'], 2));
            if ($distance < $minDistance) {
                $minDistance = $distance;
                $closestClusterId = $index;
            }
        }
        return $closestClusterId;
    }
    private function updateCentroids($data, $clusters, $k)
    {
        $newCentroids = [];
        for ($i = 0; $i < $k; $i++) {
            if (isset($clusters[$i]) && count($clusters[$i])
> 0) {
                $frekuensiTotal =
array_sum(array_column($clusters[$i], 'total_frekuensi'));
                $kerusakanTotal =
array_sum(array_column($clusters[$i], 'total_kerusakan'));
                $korbanTotal =
array_sum(array_column($clusters[$i], 'total_korban'));
                $clusterSize = count($clusters[$i]);
```

```
                $newCentroids[] = [
                    'total_frekuensi' => $frekuensiTotal /
$clusterSize,
                    'total_kerusakan' => $kerusakanTotal /
$clusterSize,
                    'total_korban' => $korbanTotal /
$clusterSize,
                ];
            } else {
                $newCentroids[] = $this->initCentroids($data,
1)[0];
            }
        }
        return $newCentroids;
    }
    private function insertIterationData($iteration,
$centroids, $clusters, $euclidean_distance, $tahun)
    {
        $labeledCentroids = [];
        $membersCount = ['C1' => 0, 'C2' => 0, 'C3' => 0];
        foreach ($centroids as $index => $centroid) {
            $label = '';
            switch ($index) {
                case 0:
                    $label = 'C1';
                    break;
                case 1:
                    $label = 'C2';
                    break;
                case 2:
                    $label = 'C3';
                    break;
            }

            if (isset($clusters[$index])) {
                $membersCount[$label] =
count($clusters[$index]);
            }

            $labeledCentroids[] = [
                'label' => $label,
                'centroid' => $centroid,
           ];
        }
        return $labeledCentroids;
    }
private function insertHasilKlasterisasi($clusters,
$currentHash)
    {
        foreach ($clusters as $index => $clusterData) {
            $clusterLabel = '';
            switch ($index) {
                case 0:
                    $clusterLabel = 'C1';  // Rendah
                    break;
                case 1:
                    $clusterLabel = 'C2';  // Sedang
                    break;
                case 2:
```

```
                    $clusterLabel = 'C3';  // Tinggi
                    break;
            }
            foreach ($clusterData as $data) {
                DB::table('tb_clustering')->insert([
                    'id_kotakab' => $data['id_kotakab'],
                    'id_kecamatan' => $data['id_kecamatan'],
                    'frekuensi_kejadian' =>
$data['total_frekuensi'],
                    'total_kerusakan' =>
$data['total_kerusakan'],
                    'total_korban' => $data['total_korban'],
                    'cluster' => $clusterLabel, // Simpan
label cluster (C1, C2, atau C3)
                    'tahun' => $data['tahun'],
                    'created_at' => now(),
                    'data_hash' => $currentHash, // Simpan
hash data
                ]);
            }
        }
    }
    private function initCentroids($data)
    {
        usort($data, function ($a, $b) {
            $sumA = $a['total_frekuensi'] +
$a['total_kerusakan'] + $a['total_korban'];
            $sumB = $b['total_frekuensi'] +
$b['total_kerusakan'] + $b['total_korban'];
            return $sumA <=> $sumB;
        });
        $centroids = [];
        $centroids[] = [
            'nama_kotakab' => $data[0]['nama_kabupaten'],
            'nama_kecamatan' => $data[0]['nama_kecamatan'],
            'total_frekuensi' => $data[0]['total_frekuensi'],
            'total_kerusakan' => $data[0]['total_kerusakan'],
            'total_korban' => $data[0]['total_korban'],
        ];
        $medianIndex = floor(count($data) / 2);
        $centroids[] = [
            'nama_kotakab' =>
$data[$medianIndex]['nama_kabupaten'],
            'nama_kecamatan' =>
$data[$medianIndex]['nama_kecamatan'],
            'total_frekuensi' =>
$data[$medianIndex]['total_frekuensi'],
            'total_kerusakan' =>
$data[$medianIndex]['total_kerusakan'],
            'total_korban' =>
$data[$medianIndex]['total_korban'],
        ];
        $lastIndex = count($data) - 1; // Mengambil indeks
terakhir
        $centroids[] = [
            'nama_kotakab' =>
$data[$lastIndex]['nama_kabupaten'],
            'nama_kecamatan' =>
$data[$lastIndex]['nama_kecamatan'],
```

```
            'total_frekuensi' =>
$data[$lastIndex]['total_frekuensi'],
            'total_kerusakan' =>
$data[$lastIndex]['total_kerusakan'],
            'total_korban' =>
$data[$lastIndex]['total_korban'],
        ];

        return $centroids;
    }
    private function assignClusters($data, $centroids)
    {
        $clusters = [];
        foreach ($data as $point) {
            $minDistance = PHP_INT_MAX;
            $closestCentroid = null;
            foreach ($centroids as $index => $centroid) {
                $distance =
sqrt(pow($point['total_frekuensi'] -
                    $centroid['total_frekuensi'], 2) +
pow($point['total_kerusakan'] -
                    $centroid['total_kerusakan'], 2) +
pow($point['total_korban'] -
                    $centroid['total_korban'], 2));
                if ($distance < $minDistance) {
                    $minDistance = $distance;
                    $closestCentroid = $index;
                }
            }
            $clusters[$closestCentroid][] = $point;
        }
        $datas = [
            'clusters' => $clusters,
            'euclidean_distances' => array_map(function
($point) use ($centroids) {
                return array_map(function ($centroid) use
($point) {
                    return sqrt(
                        pow($point['total_frekuensi'] -
$centroid['total_frekuensi'], 2) +
                            pow($point['total_kerusakan'] -
$centroid['total_kerusakan'], 2) +
                            pow($point['total_korban'] -
$centroid['total_korban'], 2)
                    );
                }, $centroids);
            }, $data)
        ];
        return $datas;
    }
    private function calculateSilhouetteScor1e($data,
$clusters, $centroids)
    {
        $calculateEuclideanDistance = function ($point1,
$point2) {
            return sqrt(pow($point1['total_frekuensi'] -
$point2['total_frekuensi'], 2) +
                pow($point1['total_kerusakan'] -
$point2['total_kerusakan'], 2) +
```

```
                pow($point1['total_korban'] -
$point2['total_korban'], 2));
        };
        $distanceMatrix = [];
        foreach ($data as $i => $point1) {
            foreach ($data as $j => $point2) {
                if (!isset($distanceMatrix[$i])) {
                    $distanceMatrix[$i] = [];
                }
                $distanceMatrix[$i][$j] =
$calculateEuclideanDistance($point1, $point2);
            }
        }
        $silhouetteScores = [];
        foreach ($data as $i => $point) {
            $clusterId = $this->getClosestCluster($point,
$centroids);
            $sameClusterPoints = $clusters[$clusterId];

            $a = 0;
            $numSameClusterPoints =
count($sameClusterPoints);
            for ($j = 0; $j < $numSameClusterPoints; $j++) {
                // Menghitung rata-rata jarak ke titik lain
dalam cluster yang sama
                if ($sameClusterPoints[$j] !== $point) {
                    $samePointIndex =
array_search($sameClusterPoints[$j], $data);
                    $a +=
$distanceMatrix[$i][$samePointIndex];
                }
            }
            $a /= ($numSameClusterPoints - 1);
            $b = PHP_INT_MAX;
            foreach ($centroids as $index => $centroid) {
                if ($index !== $clusterId) {
                    $otherClusterPoints = $clusters[$index];
                    $bClusterDistance = 0;
                    $numOtherClusterPoints =
count($otherClusterPoints);
                    foreach ($otherClusterPoints as
$otherPoint) {
                        $otherPointIndex =
array_search($otherPoint, $data);
                        $bClusterDistance +=
$distanceMatrix[$i][$otherPointIndex];
                    }
                    $bClusterDistance /=
$numOtherClusterPoints;
                    if ($bClusterDistance < $b) {
                        $b = $bClusterDistance;
                    }
                }
            }
            $silhouetteScore = ($b - $a) / max($a, $b);
            $silhouetteScores[] = $silhouetteScore;
        }
        $averageSilhouetteScore =
array_sum($silhouetteScores) / count($silhouetteScores);
```

```
        return $averageSilhouetteScore;
    }
    private function calculateSilhouetteScore($data,
$clusters, $centroids)
    {
        $calculateEuclideanDistance = function ($point1,
$point2) {
            return sqrt(pow($point1['total_frekuensi'] -
$point2['total_frekuensi'], 2) +
                pow($point1['total_kerusakan'] -
$point2['total_kerusakan'], 2) +
                pow($point1['total_korban'] -
$point2['total_korban'], 2));
        };
        $silhouetteScores = [];
        $logDetails = [];
        foreach ($data as $index => $point) {
            $clusterId = $this->getClosestCluster($point,
$centroids); // Tentukan cluster tempat titik berada
            $sameClusterPoints = $clusters[$clusterId];
            $logDetails[$index] = [
                'point' => $point,
                'clusterId' => $clusterId,
                'a' => 0,
                'b' => PHP_INT_MAX,
                'silhouetteScore' => null,
                'intraClusterDistances' => [],
                'interClusterDistances' => [],
            ];
            $a = 0;
            foreach ($sameClusterPoints as $sameIndex =>
$samePoint) {
                if ($samePoint !== $point) {
                    $distance =
$calculateEuclideanDistance($point, $samePoint);

$logDetails[$index]['intraClusterDistances'][$sameIndex] =
$distance;
                    $a += $distance;
                }
            }
            $a /= count($sameClusterPoints) - 1;
            $logDetails[$index]['a'] = $a;
            $b = PHP_INT_MAX;
            foreach ($centroids as $centroidIndex =>
$centroid) {
                if ($centroidIndex !== $clusterId) {
                    $otherClusterPoints =
$clusters[$centroidIndex];
                    $bClusterDistance = 0;
                    foreach ($otherClusterPoints as
$otherIndex => $otherPoint) {
                        $distance =
$calculateEuclideanDistance($point, $otherPoint);

$logDetails[$index]['interClusterDistances'][$otherIndex] =
$distance;
                        $bClusterDistance += $distance;
```

```
                    }
                    $bClusterDistance /=
count($otherClusterPoints);
                    if ($bClusterDistance < $b) {
                        $b = $bClusterDistance;
                    }
                }
            }
            $logDetails[$index]['b'] = $b;
            $silhouetteScore = ($b - $a) / max($a, $b);
            $logDetails[$index]['silhouetteScore'] =
$silhouetteScore;
            $silhouetteScores[] = $silhouetteScore;
        }
        $averageSilhouetteScore =
array_sum($silhouetteScores) / count($silhouetteScores);
        $logDetails['averageSilhouetteScore'] =
$averageSilhouetteScore;

        return $averageSilhouetteScore;
    }
    private function getClosestCluster($point, $centroids)
    {
        $minDistance = PHP_INT_MAX;
        $closestClusterId = null;
        foreach ($centroids as $index => $centroid) {
            $distance = sqrt(pow($point['total_frekuensi'] -
$centroid['total_frekuensi'], 2) +
                pow($point['total_kerusakan'] -
$centroid['total_kerusakan'], 2) +
                pow($point['total_korban'] -
$centroid['total_korban'], 2));

            if ($distance < $minDistance) {
                $minDistance = $distance;
                $closestClusterId = $index;
            }
        }

        return $closestClusterId;
    }
    private function updateCentroids($data, $clusters, $k)
    {
        $newCentroids = [];
        for ($i = 0; $i < $k; $i++) {
            if (isset($clusters[$i]) && count($clusters[$i])
> 0) {
                $frekuensiTotal =
array_sum(array_column($clusters[$i], 'total_frekuensi'));
                $kerusakanTotal =
array_sum(array_column($clusters[$i], 'total_kerusakan'));
                $korbanTotal =
array_sum(array_column($clusters[$i], 'total_korban'));
                $clusterSize = count($clusters[$i]);

                $newCentroids[] = [
                    'total_frekuensi' => $frekuensiTotal /
$clusterSize,
```

```
                    'total_kerusakan' => $kerusakanTotal /
$clusterSize,
                    'total_korban' => $korbanTotal /
$clusterSize,
                ];
            } else {
                $newCentroids[] = $this->initCentroids($data,
1)[0];
            }
        }
        return $newCentroids;
    }

    private function insertIterationData($iteration,
$centroids, $clusters, $euclidean_distance, $tahun)
    {
        $labeledCentroids = [];
        $membersCount = ['C1' => 0, 'C2' => 0, 'C3' => 0];
        foreach ($centroids as $index => $centroid) {
            $label = '';
            switch ($index) {
                case 0:
                    $label = 'C1';  // Rendah
                    break;
                case 1:
                    $label = 'C2';  // Sedang
                    break;
                case 2:
                    $label = 'C3';  // Tinggi
                    break;
            }
            if (isset($clusters[$index])) {
                $membersCount[$label] =
count($clusters[$index]);
            }
            $labeledCentroids[] = [
                'label' => $label,
                'centroid' => $centroid,
            ];
            DB::table('tb_log_iterasi')->insert([
                'tahun' => $tahun,
                'iteration' => $iteration,
                'cluster_label' => $label,
                'centroid_frekuensi' =>
$centroid['centroid']['total_frekuensi'],
                'centroid_kerusakan' =>
$centroid['centroid']['total_kerusakan'],
                'centroid_korban' =>
$centroid['centroid']['total_korban'],
                'created_at' => now(),
                'updated_at' => now(),
                'type' => 'centroid'
            ]);
            foreach ($clusters[$index] as $point) {
                $distanceC1 =
sqrt(pow($point['total_frekuensi'] -
$centroids[0]['centroid']['total_frekuensi'], 2) +
                    pow($point['total_kerusakan'] -
$centroids[0]['centroid']['total_kerusakan'], 2) +
```

```
                    pow($point['total_korban'] -
$centroids[0]['centroid']['total_korban'], 2));
                $distanceC2 =
sqrt(pow($point['total_frekuensi'] -
$centroids[1]['centroid']['total_frekuensi'], 2) +
                    pow($point['total_kerusakan'] -
$centroids[1]['centroid']['total_kerusakan'], 2) +
                    pow($point['total_korban'] -
$centroids[1]['centroid']['total_korban'], 2));
                $distanceC3 =
sqrt(pow($point['total_frekuensi'] -
$centroids[2]['centroid']['total_frekuensi'], 2) +
                    pow($point['total_kerusakan'] -
$centroids[2]['centroid']['total_kerusakan'], 2) +
                    pow($point['total_korban'] -
$centroids[2]['centroid']['total_korban'], 2));
                $minDistance = min($distanceC1, $distanceC2,
$distanceC3);
                $closestCluster = '';
                if ($minDistance == $distanceC1) {
                    $closestCluster = 'C1';
                } elseif ($minDistance == $distanceC2) {
                    $closestCluster = 'C2';
                } else {
                    $closestCluster = 'C3';
                }
                DB::table('tb_log_iterasi')->insert([
                    'tahun' => $tahun,
                    'iteration' => $iteration,
                    'cluster_label' => $closestCluster
                    'frekuensi_kejadian' =>
$point['total_frekuensi'],
                    'total_kerusakan' =>
$point['total_kerusakan'],
                    'total_korban' => $point['total_korban'],
                    'id_kotakab' => $point['id_kotakab'],
                    'id_kecamatan' => $point['id_kecamatan'],
                    'c1' => $distanceC1,
                    'c2' => $distanceC2,
                    'c3' => $distanceC3
                    'terdekat' => $closestCluster
                    'created_at' => now(),
                    'updated_at' => now(),
                    'type' => 'member'
                ]);
            }
        }
    }
}
```

2. Source Code PemetaanController.php

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;
class PemetaanController extends Controller
{
```

```
    public function showMap()
    {
        $tahunList = DB::table('tb_clustering')
            ->select('tahun')
            ->distinct()
            ->orderBy('tahun', 'asc')
            ->pluck('tahun');
        $geojsonFile =
file_get_contents(public_path('data/kec_jatim.geojson'));
        $geojsonData = json_decode($geojsonFile);
        $clusters = DB::table('tb_clustering')
            ->join('tb_kecamatan',
'tb_clustering.id_kecamatan', '=', 'tb_kecamatan.id')
            ->join('tb_kotakab', 'tb_kecamatan.id_kotakab',
'=', 'tb_kotakab.id')
            ->select('tb_kecamatan.id as id_kecamatan',
'tb_kotakab.id as id_kotakab', 'tb_kotakab.nama_kotakab',
'tb_kecamatan.nama_kecamatan', 'tb_clustering.cluster')
            ->get();
        $clusterMap = [];
        foreach ($clusters as $cluster) {
            // Mengubah nama_kecamatan ke uppercase
            $cleanedNamaKecamatan = strtoupper(str_replace('
', '', $cluster->nama_kecamatan));
            $clusterMap[$cleanedNamaKecamatan] = [
                'id_kotakab' => $cluster->id_kotakab,
                'nama_kotakab' => $cluster->nama_kotakab,
                'id_kecamatan' => $cluster->id_kecamatan,
                'nama_kecamatan' => $cluster->nama_kecamatan,
                'cluster' => $cluster->cluster,
            ];
        }
        Log::info("GeoJSON Features Count: ", ['count' =>
count($geojsonData->features)]);
        foreach ($geojsonData->features as $feature) {
            $nama_kecamatan = strtoupper(str_replace(' ', '',
$feature->properties->NAME_3));
            Log::info("Processing feature for: ",
['kabupaten' => $clusterMap[$nama_kecamatan]['nama_kotakab']
?? null, 'kecamatan' => $nama_kecamatan]);
            if (isset($clusterMap[$nama_kecamatan])) {
                $feature->properties->cluster =
$clusterMap[$nama_kecamatan]['cluster'];
                Log::info("Assigned cluster: ", ['cluster' =>
$feature->properties->cluster]);
            } else {
                Log::info("Cluster not found for: ",
['kabupaten' => $clusterMap[$nama_kecamatan]['nama_kotakab']
?? null, 'kecamatan' => $nama_kecamatan]);
                $feature->properties->cluster = null;
            }
        }
        $modifiedGeojsonString = json_encode($geojsonData);

        return view('map.index', ['geojson' =>
$modifiedGeojsonString, 'tahunList' => $tahunList]);
    }
    public function filterByYear(Request $request)
    {
```

```
        $geojsonFile =
file_get_contents(public_path('data/kec_jatim.geojson'));
        $geojsonData = json_decode($geojsonFile);
        $year = $request->input('year');
        $clusters = DB::table('tb_clustering')
            ->join('tb_kecamatan',
'tb_clustering.id_kecamatan', '=', 'tb_kecamatan.id')
            ->join('tb_kotakab', 'tb_kecamatan.id_kotakab',
'=', 'tb_kotakab.id')
            ->select('tb_kecamatan.id as id_kecamatan',
'tb_kotakab.id as id_kotakab', 'tb_kotakab.nama_kotakab',
'tb_kecamatan.nama_kecamatan', 'tb_clustering.cluster');
        if ($year) {
            $clusters = $clusters-
>where('tb_clustering.tahun', $year);
        }
        $clusters = $clusters->get();
        $clusterMap = [];
        foreach ($clusters as $cluster) {
            $cleanedNamaKecamatan = strtoupper(str_replace('
', '', $cluster->nama_kecamatan));
            $clusterMap[$cleanedNamaKecamatan] = [
                'id_kotakab' => $cluster->id_kotakab,
                'nama_kotakab' => $cluster->nama_kotakab,
                'id_kecamatan' => $cluster->id_kecamatan,
                'nama_kecamatan' => $cluster->nama_kecamatan,
                'cluster' => $cluster->cluster,
            ];
        }
        Log::info("GeoJSON Features Count: ", ['count' =>
count($geojsonData->features)]);
        foreach ($geojsonData->features as $feature) {
            $nama_kecamatan = strtoupper(str_replace(' ', '',
$feature->properties->NAME_3));
            Log::info("Processing feature for: ",
['kabupaten' => $clusterMap[$nama_kecamatan]['nama_kotakab']
?? null, 'kecamatan' => $nama_kecamatan]);
            if (isset($clusterMap[$nama_kecamatan])) {
                $feature->properties->cluster =
$clusterMap[$nama_kecamatan]['cluster'];
                Log::info("Assigned cluster: ", ['cluster' =>
$feature->properties->cluster]);
            } else {
                Log::info("Cluster not found for: ",
['kabupaten' => $clusterMap[$nama_kecamatan]['nama_kotakab']
?? null, 'kecamatan' => $nama_kecamatan]);
                $feature->properties->cluster = null;
            }
        }
        $modifiedGeojsonString = json_encode($geojsonData);

        return response()->json($geojsonData);
    }
}
```