DAFTAR PUSTAKA

Andraud, M., Bougeard, S., Chesnoiu, T., & Rose, N. (2021). Spatiotemporal clustering and Random Forest models to identify risk factors of African swine fever outbreak in Romania in 2018–2019. *Scientific Reports*, *11*(1), 1–12. https://doi.org/10.1038/s41598-021-81329-x

Baeda, A. Y., & Husain, F. (2012). Kajian Potensi Tsunami Akibat Gempa Bumi Bawah Laut di Perairan Pulau Sulawesi. *Jurnal Teknik Sipil*, *19*(1), 75. https://doi.org/10.5614/jts.2012.19.1.7

Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data and Knowledge Engineering*, *60*(1), 208–221. https://doi.org/10.1016/j.datak.2006.01.013

Creţulescu, R. G., Morariu, D. I., Breazu, M., & Volovici, D. (2019). DBSCAN Algorithm for Document Clustering. *International Journal of Advanced Statistics and IT&C for Economics and Life Sciences*, *9*(1), 58–66. https://doi.org/10.2478/ijasitels-2019-0007

Dahmouni, A., El Moutaouakil, K., & Satori, K. (2018). Clustering and jarque-bera normality test to face recognition. *Procedia Computer Science*, *127*, 246–255. https://doi.org/10.1016/j.procs.2018.01.120

Faraouk, K. El, Witriyono, H., Deslianti, D., & Veronika, N. D. M. (2023). ST-DBSCAN Algorithm Implementation At Riau Province Forest Fire Points (2015-2022). *Jurnal Komputer, Informasi Dan Teknologi (JKOMITEK)*, *3*(1), 97–104. https://doi.org/10.53697/jkomitek.v3i1.1191

Gaonkar, N. M., & Sawant, K. (2013). AutoEpsDBSCAN : DBSCAN with Eps Automatic for Large Dataset. *International Journal on Advanced Computer Theory and Engineering(IJACTE)*, *2*(2), 2319–2526.

Handoko, K., & Lesmana, L. S. (2018). Data Mining Pada Jumlah Penumpang Menggunakan Metode Clustering. *Snistek*, *1*, 97–102.

Huang, X., Ma, T., Liu, C., & Liu, S. (2023). GriT-DBSCAN: A spatial clustering algorithm for very large databases. *Pattern Recognition*, *142*. https://doi.org/10.1016/j.patcog.2023.109658

Ihda, E., Sudarsono, B., & Awaluddin, M. (2015). Analisis Deformasi Seismik Sesar Matano Menggunakan Gnss Dan Interferometrik Sar. *Jurnal Geodesi Undip*, *4*(April), 86–94. https://ejournal3.undip.ac.id/index.php/geodesi/article/viewFile/27163/23790

Indrawan, N. A., & Adrianto, H. A. (2014). Spatio - Temporal Clustering Hotspot di Sumatera Selatan Tahun 2002 - 2003 Menggunakan Algoritme ST - DBSCAN dan Bahasa Pemrograman R Spatio - Temporal Clustering Hotspot in South Sumatera From 2002 - 2003 Using ST - DBSCAN Algorithm and R Programming Lan. *Jurnal Ilmu Komputer Dan Agri-Informatika*, *3*, 112–121.

Indrawan, N. A., & Adrianto, H. A. (2016). Spatio-Temporal Clustering Hotspot di Sumatera Selatan Tahun 2002-2003 Menggunakan Algoritme ST-DBSCAN dan Bahasa Pemrograman R. *Jurnal Ilmu Komputer Dan Agri-Informatika*, *3*(2), 112. https://doi.org/10.29244/jika.3.2.112-121

Latifi-Pakdehi, A., & Daneshpour, N. (2021). DBHC: A DBSCAN-based hierarchical clustering algorithm. *Data and Knowledge Engineering*, *135*(April), 101922. https://doi.org/10.1016/j.datak.2021.101922

Liu, B., Xu, G., Xu, Q., & Zhang, N. (2012). Outlier Detection Data Mining of Tax Based on Klaster. *Physics Procedia*, *33*, 1689–1694. https://doi.org/10.1016/j.phpro.2012.05.272

Manamperi, A., & Manamperi, L. A. N. (2023). *The Role of Statistics in Data Mining*. *December*. https://www.researchgate.net/publication/376951359

Nahdliyah, M. A., Widiharih, T., & Prahutama, A. (2019). METODE k-MEDOIDS CLUSTERING DENGAN VALIDASI SILHOUETTE INDEX DAN C-INDEX (Studi Kasus Jumlah Kriminalitas Kabupaten/Kota di Jawa Tengah Tahun 2018). *Jurnal Gaussian*, *8*(2), 161–170. https://doi.org/10.14710/j.gauss.v8i2.26640

Pasau, G., -, F., & Tamuntuan, G. H. (2017). Pengamatan Seismisitas Gempa Bumi Di Wilayah Pulau Sulawesi Menggunakan Perubahan Nilai a-b. *Jurnal MIPA*, *6*(1), 31. https://doi.org/10.35799/jm.6.1.2017.15988

Pölitz, C., & Andrienko, G. N. (2010). Finding arbitrary shaped klasters with related extents in space and time. *1st International Symposium on Visual Analytics Science and Technology, EuroVAST@EuroVis 2010*, 19–25. https://doi.org/10.2312/PE/EuroVAST/EuroVAST10/019-025

Sharma, A., & Sharma, A. (2017). KNN-DBSCAN: Using k-nearest neighbor information for parameter-free density based clustering. *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies, ICICICT 2017*, *2018-Janua*, 787–792. https://doi.org/10.1109/ICICICT1.2017.8342664

Susanta, F. F., Pratama, C., Aditya, T., Khomaini, A. F., & Abdillah, H. W. K. (2019). G*eovisual Analytics of Spatio-Temporal Earthquake Data in Indonesia*. *JGISE: Journal of Geospatial Information Science and Engineering*, *2*(2), 185–194. https://doi.org/10.22146/jgise.51131

Trisminingsih, R., & Shaztika, S. S. (2017). ST-DBSCAN clustering module in SpagoBI for hotspots distribution in Indonesia. *Proceedings - 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering, ICITACEE 2016*, 327–330. https://doi.org/10.1109/ICITACEE.2016.7892465

LAMPIRAN

Lampiran 1. Data Gempa Bumi di Pulau Sulawesi Tahun 2021-2023

No	Waktu Kejadian	Latitude	Longitude	Depth	Mag
1	1-Jan-21	-0.1874	124.442	61.99	4.5
2	2-Jan-21	0.4377	121.8602	121.69	4.2
3	2-Jan-21	-1.9263	122.549	10	4.4
4	3-Jan-21	-2.7349	122.1901	10	4.5
5	3-Jan-21	-2.7566	122.2402	10	5
6	6-Jan-21	0.0658	122.9487	148	6.1
7	11-Jan-21	1.0276	119.9894	9.85	5.2
8	14-Jan-21	-3.0171	118.9502	10	5.2
9	15-Jan-21	-2.9335	118.9545	10	4.9
10	15-Jan-21	1.5145	125.9643	95.6	5
11	17-Jan-21	0.6734	124.3601	128.22	4.2
12	17-Jan-21	-0.2674	125.071	73.12	4.4
13	19-Jan-21	-0.0023	120.0833	79.29	4.8
14	21-Jan-21	-2.9966	118.9163	10	4.2
15	21-Jan-21	-2.768	122.2842	10	4
16	21-Jan-21	-2.7803	122.4805	10	4.1
17	22-Jan-21	0.3979	122.2345	187.17	4.2
18	22-Jan-21	0.5497	124.5404	10	3.9
19	27-Jan-21	0.4146	123.7554	202.93	4.3
20	27-Jan-21	0.0083	123.4695	137.17	5
21	30-Jan-21	1.5398	121.9731	480.99	4.5
22	30-Jan-21	0.5312	125.5284	57.12	5
23	31-Jan-21	-3.0174	118.9466	10	4.4
24	3-Feb-21	-3.0035	118.9307	10	4.9
25	4-Feb-21	2.0533	125.4863	10	4.1
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
441	16-Dec-23	-0.5444	122.2196	10	4.2
442	20-Dec-23	0.2783	120.4221	75.66	4
443	20-Dec-23	-0.0928	123.0929	141.084	4.5
444	21-Dec-23	1.2821	123.142	70.02	4.7
445	23-Dec-23	0.4641	125.3686	56.326	4.9
446	25-Dec-23	-0.721	121.4377	10	4.3
447	25-Dec-23	0.3607	123.1898	230.316	4.6
448	26-Dec-23	-0.0357	122.9301	133.502	4.5

Lampiran 2. Jarak Euclidean Spasial

	1	2	3	4	5	6	7	8	9	10	11	12	13	⋯	⋯	448
1	0	2.656	2.570	3.400	3.383	1.514	4.615	6.177	6.136	2.283	0.864	0.634	4.362	…	…	1.519
2	2.656	0	2.462	3.189	3.216	1.150	1.961	4.517	4.450	4.243	2.510	3.287	1.830	⋯	⋯	1.169
3	2.570	2.462	0	0.884	0.885	2.031	3.908	3.760	3.732	4.848	3.168	3.018	3.127	⋯	⋯	1.928
4	3.400	3.189	0.884	0	0.054	2.901	4.358	3.252	3.241	5.683	4.040	3.793	3.450	⋯	⋯	2.798
5	3.383	3.216	0.885	0.054	0	2.909	4.402	3.300	3.290	5.666	4.032	3.769	3.498	…	…	2.807
6	1.514	1.150	2.031	2.901	2.909	0	3.111	5.048	4.994	3.345	1.536	2.148	2.866	⋯	⋯	0.103
7	4.615	1.961	3.908	4.358	4.402	3.111	0	4.176	4.094	5.994	4.385	5.244	1.034	…	…	3.127
8	6.177	4.517	3.760	3.252	3.300	5.048	4.17	0	0.083	8.350	6.548	6.710	3.220	⋯	⋯	4.972
9	6.136	4.450	3.732	3.241	3.290	4.994	4.094	0.083	0	8.301	6.498	6.672	3.141	⋯	⋯	4.919
10	2.283	4.243	4.848	5.683	5.666	3.345	5.994	8.350	8.301	0	1.811	1.993	6.073	⋯	⋯	3.407
11	0.864	2.510	3.168	4.040	4.032	1.536	4.385	6.548	6.498	1.811	0	1.179	4.329	⋯	⋯	1.596
12	0.634	3.287	3.018	3.793	3.769	2.148	5.244	6.710	6.672	1.993	1.179	0	4.994	⋯	⋯	2.153
13	4.362	1.830	3.127	3.450	3.498	2.866	1.034	3.220	3.141	6.073	4.329	4.994	0	⋯	⋯	1.021
14	6.198	4.523	3.787	3.284	3.332	5.063	4.164	0.039	0.073	8.368	6.565	6.732	3.213	…	…	2.846
15	3.363	3.233	0.882	0.099	0.045	2.910	4.435	3.343	3.333	5.646	4.019	3.744	3.534	…	…	4.987
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
448	1.519	1.169	1.928	2.798	2.807	0.103	3.127	4.919	3.407	1.596	2.153	2.846	4.987	…	…	0

Lampiran 3. Jarak Euclidean Temporal

	1	2	3	4	5	6	7	8	9	10	11	12	13	…	…	448
1	0	1	1	2	2	5	10	13	14	14	16	16	18	…	…	1089
2	1	0	0	1	1	4	9	12	13	13	15	15	17	…	…	1088
3	1	0	0	1	1	4	9	12	13	13	15	15	17	…	…	1088
4	2	1	1	0	0	3	8	11	12	12	14	14	16	…	…	1087
5	2	1	1	0	0	3	8	11	12	12	14	14	16	…	…	1087
6	5	4	4	3	3	0	5	8	9	9	11	11	13	…	…	1084
7	10	9	9	8	8	5	0	3	4	4	6	6	8	…	…	1079
8	13	12	12	11	11	8	3	0	1	1	3	3	5	…	…	1076
9	14	13	13	12	12	9	4	1	0	0	2	2	4	…	…	1075
10	14	13	13	12	12	9	4	1	0	0	2	2	4	…	…	1075
11	16	15	15	14	14	11	6	3	2	2	0	0	2	…	…	1073
12	16	15	15	14	14	11	6	3	2	2	0	0	2	…	…	1073
13	18	17	17	16	16	13	8	5	4	4	2	2	0	…	…	1071
14	20	19	19	18	18	15	10	7	6	6	4	4	2	…	…	1069
15	20	19	19	18	18	15	10	7	6	6	4	4	2	…	…	1069
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
448	1051	1050	1050	1049	1049	1046	1041	1038	1037	1037	1035	1035	1033	…	…	0

Lampiran 4. Jarak Rata- Rata Terdekat Observasi

	1	2	3	4	5	6	7	8	9	10	11	12	13	...	...	448
1		13501.62	1383 7.68	1388 4.72	1389 2.6	1366 3.96	1322 7.79	1355 5.32	1354 6.86	1383 7.3	1375 3.31	1393 6.8	1335 2.89	...	...	1367 3.13
2	13858.2		1384 0.41	1388 8.66	1389 6.58	1366 3.77	1322 6.28	1355 9.54	1355 0.95	1383 4.97	1375 2.23	1393 7.09	1335 2.8	...	...	1367 3.09
3	13854.8	13501.17		1387 1.01	1387 8.78	1366 1.92	1322 9.78	1354 1.12	1353 3.01	1384 1.06	1375 3.67	1393 3.17	1335 0.81	...	...	1367 0.68
4	13851.92	13499.74	1382 1.2		1387 0.97	1365 9.68	1322 9.6	1353 3.29	1352 5.34	1384 1.43	1375 2.5	1393 0.06	1334 8.69	...	...	1366 8.24
5	13851.83	13499.69	1382 1.02	1386 3.03		1365 9.61	1322 9.58	1353 3.07	1352 5.12	1384 1.43	1375 2.45	1392 9.97	1334 8.63	...	...	1366 8.16
6	13858.16	13501.38	1383 8.85	1388 6.38	1389 4.28		1322 7.23	1355 7.09	1354 8.57	1383 6.42	1375 2.93	1393 6.98	1335 2.91	...	...	1367 3.18
7	13857.88	13499.8	1384 2.51	1389 1.89	1389 9.84	1366 3.13		1356 3.08	1355 4.38	1383 2.3	1375 0.74	1393 6.87	1335 2.32	...	...	1367 2.59
8	13850.7	13499.06	1381 8.84	1386 0.34	1386 8.04	1365 8.7	1322 9.37		1352 2.48	1384 1.35	1375 1.89	1392 8.77	1334 7.78	...	...	1366 7.2
9	13851.07	13499.27	1381 9.55	1386 1.21	1386 8.92	1365 9	1322 9.45	1353 1.25		1384 1.39	1375 2.08	1392 9.16	1334 8.06	...	...	1366 7.52
10	13857.27	13498.59	1384 3.89	1389 4.21	1390 2.18	1366 2.28	1322 2.7	1356 5.69	1355 6.9		1374 9.18	1393 6.33	1335 1.63	...	...	1367 1.85
11	13858.13	13500.51	1384 1.3	1389 0.01	1389 7.94	1366 3.57	1322 5.6	1356 1.01	1355 2.37	1383 3.96		1393 7.06	1335 2.66	...	...	1367 2.94
12	13857.97	13501.68	1383 7.29	1388 4.18	1389 2.06	1366 3.95	1322 7.95	1355 4.75	1354 6.3	1383 7.57	1375 3.41		1335 2.87	...	...	1367 3.11
13	13858.13	13501.45	1383 8.54	1388 5.95	1389 3.84	1366 3.95	1322 7.39	1355 6.62	1354 8.12	1383 6.67	1375 3.04	1393 6.94		...	...	1367 3.17
14	13850.8	13499.11	1381 9.02	1386 0.55	1386 8.26	1365 8.78	1322 9.39	1353 0.59	1352 2.69	1384 1.36	1375 1.94	1392 8.87	1334 7.85	...	...	1366 8.12
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
448	13858.12	13501.49	1383 8.39	1388 5.73	1389 3.62	1366 3.96	1322 7.46	1355 6.39	1354 7.89	1383 6.79	1375 3.1	1393 6.92	1335 2.91	...	...	0

Lampiran 5. Parameter Hasil Simulasi

No	MinPts	Eps1	Eps2	Cluster	Noise	Silhouette Coefficient	DBI	Calinski Harabahz
1	3	1	7	8	404	-0,736	21,704	3,742
2	3	1	14	8	404	-0,736	21,704	3,742
3	3	1	21	8	404	-0,736	21,704	3,742
4	3	1	28	8	404	-0,736	21,704	3,742
5	3	4	7	60	74	0,341	13,493	31,062
6	3	4	14	60	74	0,341	13,493	31,062
7	3	4	21	60	74	0,341	13,493	31,062
8	3	4	28	60	74	0,341	13,493	31,062
9	3	7	7	30	9	0,467	5,112	587,464
10	3	7	14	30	9	0,467	5,112	587,464
11	3	7	21	30	9	0,467	5,112	587,464
12	3	7	28	30	9	0,467	5,112	587,464
13	3	10	7	12	4	0,440	1,303	2155,773
14	3	10	14	12	4	0,440	1,303	2155,773
15	3	10	21	12	4	0,440	1,303	2155,773
16	3	10	28	12	4	0,440	1,303	2155,773
17	3	13	7	6	0	0,505	0,436	1148,569
18	3	13	14	6	0	0,505	0,436	1148,569
19	3	13	21	6	0	0,505	0,436	1148,569
20	3	13	28	6	0	0,505	0,436	1148,569
21	3	16	7	2	0	0,586	0,482	886,470
22	3	16	14	2	0	0,586	0,482	886,470
23	3	16	21	2	0	0,586	0,482	886,470
24	3	16	28	2	0	0,586	0,482	886,470
25	3	19	7	1	0	NA	NA	NA
26	3	19	14	1	0	NA	NA	NA
27	3	19	21	1	0	NA	NA	NA
28	3	19	28	1	0	NA	NA	NA
29	4	1	7	3	419	-0,661	15,479	5,106
30	4	1	14	3	419	-0,661	15,479	5,106
31	4	1	21	3	419	-0,661	15,479	5,106
32	4	1	28	3	419	-0,661	15,479	5,106
33	4	4	7	41	155	0,091	4,977	14,710
34	4	4	14	41	155	0,091	4,977	14,710
35	4	4	21	41	155	0,091	4,977	14,710

36	4	4	28	41	155	0,091	4,977	14,710
37	4	7	7	28	30	0,443	2,322	397,264
38	4	7	14	28	30	0,443	2,322	397,264
39	4	7	21	28	30	0,443	2,322	397,264
40	4	7	28	28	30	0,443	2,322	397,264
41	4	10	7	12	5	0,438	25,155	1362,956
42	4	10	14	12	5	0,438	25,155	1362,956
43	4	10	21	12	5	0,438	25,155	1362,956
44	4	10	28	12	5	0,438	25,155	1362,956
45	4	13	7	6	1	0,493	0,422	955,081
46	4	13	14	6	1	0,493	0,422	955,081
47	4	13	21	6	1	0,493	0,422	955,081
48	4	13	28	6	1	0,493	0,422	955,081
49	4	16	7	2	0	0,586	0,482	886,470
50	4	16	14	2	0	0,586	0,482	886,470
51	4	16	21	2	0	0,586	0,482	886,470
52	4	16	28	2	0	0,586	0,482	886,470
53	4	19	7	4	1	NA	NA	NA
54	4	19	14	4	1	NA	NA	NA
55	4	19	21	4	1	NA	NA	NA
56	4	19	28	4	1	NA	NA	NA
57	5	1	7	1	427	NA	NA	NA
58	5	1	14	1	427	NA	NA	NA
59	5	1	21	1	427	NA	NA	NA
60	5	1	28	1	427	NA	NA	NA
61	5	4	7	29	220	-0,111	8,660	13,524
62	5	4	14	29	220	-0,111	8,660	13,524
63	5	4	21	29	220	-0,111	8,660	13,524
64	5	4	28	29	220	-0,111	8,660	13,524
65	5	7	7	27	51	0,397	2,403	176,287
66	5	7	14	27	51	0,397	2,403	176,287
67	5	7	21	27	51	0,397	2,403	176,287
68	5	7	28	27	51	0,397	2,403	176,287
69	5	10	7	18	11	0,371	3,905	1143,977
70	5	10	14	18	11	0,371	3,905	1143,977
71	5	10	21	18	11	0,371	3,905	1143,977
72	5	10	28	18	11	0,371	3,905	1143,977
73	5	13	7	7	1	0,470	0,432	823,260

74	5	13	14	7	1	0,470	0,432	823,260
75	5	13	21	7	1	0,470	0,432	823,260
76	5	13	28	7	1	0,470	0,432	823,260
77	5	16	7	5	0	0,361	0,432	258,641
78	5	16	14	5	0	0,361	0,432	258,641
79	5	16	21	5	0	0,361	0,432	258,641
80	5	16	28	5	0	0,361	0,432	258,641
81	5	19	7	1	0	NA	NA	NA
82	5	19	14	1	0	NA	NA	NA
83	5	19	21	1	0	NA	NA	NA
84	5	19	28	1	0	NA	NA	NA
85	6	1	7	1	427	NA	NA	NA
86	6	1	14	1	427	NA	NA	NA
87	6	1	21	1	427	NA	NA	NA
88	6	1	28	1	427	NA	NA	NA
89	6	4	7	21	270	-0,270	3,052	12,193
90	6	4	14	21	270	-0,270	3,052	12,193
91	6	4	21	21	270	-0,270	3,052	12,193
92	6	4	28	21	270	-0,270	3,052	12,193
93	6	7	7	22	105	0,246	2,621	115,850
94	6	7	14	22	105	0,246	2,621	115,850
95	6	7	21	22	105	0,246	2,621	115,850
96	6	7	28	22	105	0,246	2,621	115,850
97	6	10	7	15	30	0,362	26,911	478,288
98	6	10	14	16	24	0,370	3,291	535,065
99	6	10	21	16	24	0,370	3,291	535,065
100	6	10	28	16	24	0,370	3,291	535,065
101	6	13	7	9	2	0,480	0,447	651,049
102	6	13	14	9	2	0,480	0,447	651,049
103	6	13	21	9	2	0,480	0,447	651,049
104	6	13	28	9	2	0,480	0,447	651,049
105	6	16	7	5	0	0,361	0,432	258,641
106	6	16	14	5	0	0,361	0,432	258,641
107	6	16	21	5	0	0,361	0,432	258,641
108	6	16	28	5	0	0,361	0,432	258,641
109	6	19	7	1	0	NA	NA	NA
110	6	19	14	1	0	NA	NA	NA
111	6	19	21	1	0	NA	NA	NA

112	6	19	28	1	0	NA	NA	NA

Lampiran 8. Script RStudio

```
#Package
library(cluster)
library(fpc) #DBSCAN Algorithm
library(sp)
library(ggplot2)
library(maptools)
library(rgdal)
library(plyr)
library(svglite)
library(readxl)
library(dbscan) #fungsi dbscan
library(rgl) #fungsi plot (?)
library(factoextra)
library(ggmap)
library(spatstat)
# Data
DATA <- read_excel("D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/DATARANI.
xlsx")
DATA
# read shapefile
map <- readOGR ("D:/AA TESIS FIX/GEMPA/Pulau Sulawesi (Kabupaten)/Ka
bupaten_Pulau_Sulawesi.shp")
#Visualisasi Titik Gempa
#convert to dataframe
maps <- fortify(map)
mergemap <- join(maps, map@data, by="id")
ggplot(mergemap) +
  aes(long, lat, group = group) +
  geom_polygon(data = map, aes(long, lat, group = group), color = "r
ed", fill = "black") + geom_path(data = map, aes(long, lat, group =
group), color = "white") + geom_polygon(data = mergemap, aes(long, l
at), fill = "black", color = "black") + theme(legend.position = "non
e") +
  geom_point(data = DATA, aes(Longitude, Latitude, group = 1), size
= 1.3, color = "red") + xlab("Latitude") +
  ylab("Longitude") +
  coord_equal()
```

```
# Visualisasi Kedalaman Gempa
#convert to dataframe
maps <- fortify(map)
mergemap <- join(maps, map@data, by="id")
ggplot(mergemap) +
  aes(long, lat, group = group) +
  geom_polygon(data = map, aes(long, lat, group = group), color = "g
rey") +
  geom_path(data = map, color = "white") +
  geom_polygon(data = mergemap, aes(long, lat)) +
  theme(legend.position = "right") +
  geom_point(data = DATA, aes(Longitude, Latitude, group = 1, color
= Depth), size = 1.3) +
  scale_color_gradientn(colors = c("darkred", "red", "orange", "yell
ow", "white")) +
  xlab("Latitude") + ylab("Longitude") +
  coord_equal()
# Visualisasi Kekuatan Gempa
#convert to dataframe
maps <- fortify(map)
## Regions defined for each Polygons
mergemap <- join(maps, map@data, by="id")
ggplot(mergemap) +
  aes(long, lat, group = group) +
  geom_polygon(data = map, aes(long, lat, group = group), color = "g
rey") +
  geom_path(data = map, color = "white") +
  geom_polygon(data = mergemap, aes(long, lat)) +
  theme(legend.position = "right") +
  geom_point(data = DATA, aes(Longitude, Latitude, group = 1, color
= Mag), size = 1.3) +
  scale_color_gradientn(colors = c("darkred", "red", "orange", "yell
ow", "white")) +
  xlab("Latitude") + ylab("Longitude") +
  coord_equal()
#Menghitung jarak euclidean
#Euclidean Spasial
d = dist(cbind(DATA$Latitude,DATA$Longitude),"euclidean")
View(as.matrix(d))
write.csv(as.matrix(d), "D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/Eucl
idean Spasiall.csv")
#Euclidean Temporal
```

```
e= dist(DATA$Code,"euclidean")
View(as.matrix(e))
write.csv(as.matrix(e), "D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/Eucl
idean Temporall.csv")
#PENENTIAN EPS
#Membuat plot  parameter "Time, Latitude, Longitude"
kNNdistplot(DATA[,4:5], k = 3)
#K=kNNdistplot(DATA[,4:6], k = 3)
#write.csv(as.matrix(K), "D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/KNN
3.csv")
abline(h= 1, col = "red", lty = 3)
abline(h= 9, col = "red", lty = 3)
kNNdistplot(DATA[,4:5], k = 4)
#L=kNNdistplot(DATA[,4:6], k = 4)
#write.csv(as.matrix(L), "D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/KNN
4.csv")
abline(h= 1, col = "red", lty = 3)
abline(h= 11, col = "red", lty = 3)
kNNdistplot(DATA[,4:5], k = 5)
#M=kNNdistplot(DATA[,4:6], k = 5)
#write.csv(as.matrix(M), "D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/KNN
5.csv")
abline(h= 2  , col = "red", lty = 3)
abline(h= 15, col = "red", lty = 3)
kNNdistplot(DATA[,4:5], k = 6)
#N=kNNdistplot(DATA[,4:6], k = 6)
#write.csv(as.matrix(N), "D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/KNN
6.csv")
abline(h=  3, col = "red", lty = 3)
abline(h= 18, col = "red", lty = 3)
# Kode program ST-DBSCAN
stdbscan = function(datafile, ε_1, eps2, minpts, seeds=TRUE, countmo
de=1:nrow(datafile)) {
  data_spasial<- dist(cbind(datafile$Latitude, datafile$Longitude))
  data_temporal<- dist(datafile$Code)
  data_spasial <- as.matrix(data_spasial)
  data_temporal <- as.matrix(data_temporal)
  n<- nrow(data_spasial)
  classn <- cv <- integer(n)
  isseed <- logical (n)
  cn     <- integer (1)
  for (i in 1:n) {
```

```
    if (i %in% countmode)
      unclass<- (1:n) [cv < 1]
    if (cv[i] == 0) {
      #mencari tetangga spasial dan temporal
      ddreachables <- intersect (unclass [data_spasial [i, unclass] <
= ε_1],
                                 unclass [data_temporal [i, unclass]
<= eps2])
      if (length(ddreachables) + classn[i] < minpts)
        cv[1] <-(-1)
      else {
        cn <- cn + 1
        cv[i] <- cn
        isseed[i] <- TRUE
        ddreachables <- setdiff (ddreachables, i)
        unclass<- setdiff(unclass, i)
        classn[ddreachables] <- classn[ddreachables] + 1
        while (length(ddreachables)) {
          cv[ddreachables] <- cn
          ap <- ddreachables
          ddreachables <- integer()
          for (i2 in seq(along=ap)) {
            j <- ap[i2]
            jreachables <- intersect (unclass [data_spasial [j, uncl
ass] <= ε_1],
                                      unclass [data_temporal[j, uncl
ass] <= eps2])
            if (length(jreachables)+ classn[j] >= minpts) {
              isseed[j] <- TRUE
              cv[jreachables[cv[jreachables] < 0]] <- cn
              ddreachables <- union (ddreachables, jreachables[cv[jr
eachables] == 0])}
            classn[jreachables] <- classn[jreachables] + 1
            unclass <- setdiff(unclass, j)
          }
        }
      }
    }
    if (!length(unclass))
      break
  }
  rm(class)
```

```
  if (any (cv == (-1))) {
    cv[cv == (-1)] <- 0 }

  out <- list (cluster = cv, ε_1 = ε_1, eps2=eps2, minpts = minpts)
  class (out) <- "stdbscan"
  out
}
#ANALISIS PERIODE 1-27
# Data
PERIODE1 <- read_excel("D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/PERIO
DE/1-27.xlsx")
############## Memastikan struktur data MAGMAX
##names(MAGMAX)
# read shapefile
map <- readOGR("D:/AA TESIS FIX/GEMPA/Pulau Sulawesi (Kabupaten)/Kab
upaten_Pulau_Sulawesi.shp")
map@data$id <- rownames(map@data)
```

Lampiran 9. Script Python

```
#########Package
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from st_dbscan import ST_DBSCAN
from datetime import timedelta
from sklearn import metrics
from scipy.spatial import cKDTree
import matplotlib.pyplot as plt
#IMPORT DATA#
data = pd.read_csv('D:/AAA BISMILLAH TESIS/PERSIAPAN
UJIAN/DATARANI.csv')
data.head()
########### JARAK TETANGGA TERDEKAT
# Pastikan untuk menggunakan tanda kutip ganda atau tunggal yang benar
file_path = "D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/NNA/NNA.csv"
data = pd.read_csv(file_path)
# Function to calculate the distance between two points given their
latitude and longitude
def haversine(lat1, lon1, lat2, lon2):
    R = 6371  # Radius of the Earth in kilometers
    dlat = np.radians(lat2 - lat1)
    dlon = np.radians(lat2 - lon2)
    a = np.sin(dlat / 2) ** 2 + np.cos(np.radians(lat1)) *
np.cos(np.radians(lat2)) * np.sin(dlon / 2) ** 2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))
    distance = R * c
    return distance
# Calculate the distance matrix
def calculate_distance_matrix(data):
    num_points = len(data)
    distance_matrix = np.zeros((num_points, num_points))
    for i in range(num_points):
        for j in range(num_points):
            if i != j:
```

```
                distance_matrix[i][j]                                    =
haversine(data.iloc[i]['Latitude'],       data.iloc[i]['Longitude'],
data.iloc[j]['Latitude'], data.iloc[j]['Longitude'])
            else:
                distance_matrix[i][j] = np.inf  # Assign infinity to
the diagonal (distance to self)
    return distance_matrix
# Calculate the distance matrix
distance_matrix = calculate_distance_matrix(data)
# Convert the distance matrix to a DataFrame for better readability
distance_matrix_df = pd.DataFrame(distance_matrix)
# Display the distance matrix
print(distance_matrix_df)
# Load data
############## NEAREST NEIGHBORD ANALYSIS
data     =     pd.read_csv('D:/AAA     BISMILLAH     TESIS/PERSIAPAN
UJIAN/DATARANI.csv')
# Extract coordinates
coords = data[['Latitude', 'Longitude']].values
# Build KDTree for fast nearest neighbor search
tree = cKDTree(coords)
# Find nearest neighbor distances
distances, _ = tree.query(coords, k=2)
nearest_neighbor_distances = distances[:, 1]
# Calculate mean nearest neighbor distance
mean_nearest_neighbor_distance = nearest_neighbor_distances.mean()
# Calculate standard deviation of nearest neighbor distances
std_nearest_neighbor_distance = nearest_neighbor_distances.std()
# Calculate standard error of the mean nearest neighbor distance
n_points = len(coords)
sem_nearest_neighbor_distance   =   std_nearest_neighbor_distance   /
np.sqrt(n_points)
# Output results
print(f'Mean           Nearest           Neighbor           Distance:
{mean_nearest_neighbor_distance}')
print(f'Standard   Deviation   of   Nearest   Neighbor   Distances:
{std_nearest_neighbor_distance}')
print(f'Standard   Error   of   Mean   Nearest   Neighbor   Distance:
{sem_nearest_neighbor_distance}')
########## PEMBENTUKAN CLUSTER ST-DBSCAN
GEMPA = pd.DataFrame()
GEMPA['latitude'] = data['Latitude']
```

```
GEMPA['longitude'] = data['Longitude']
GEMPA['time'] = data['Code ']
import math
x = GEMPA['latitude']
y = GEMPA['longitude']
jarak = []
for i in range(len(x)):
    for j in range(i+1, len(x)):
        jarak.append(math.sqrt((x[j] - x[i])**2 + (y[j] - y[i])**2))
# Buat kumpulan kombinasi parameter
ε_11 = np.array([1,4,7,10,13,16,19])
eps21 = np.array([7,14,21,28])
minpts= np.array([3,4,5,6])
# Initialize result lists
E_1 = []
Eps2 = []
MinPts = []
clusters = []
noises = []
silhouttes = []
dbi_scores = []
ch_scores = []
# Fit clusters for all parameter combinations
for i in range(len(ε_11)):
    for j in range(len(eps21)):
        for k in range(len(minpts)):
            E_1.append(ε_11[i])
            Eps2.append(eps21[j])
            MinPts.append(minpts[k])
            cluster = ST_DBSCAN(ε_1=ε_11[i], eps2=eps21[j],
min_samples=minpts[k])
            cluster.fit(GEMPA)
            labels = cluster.labels
            n_clusters = len(set(cluster.labels)) - (1 if -1 in
labels else 0)
            n_noise = list(cluster.labels).count(-1)
            clusters.append(n_clusters)
            noises.append(n_noise)
            if n_clusters > 1:  # Only compute silhouette score if
there are at least 2 clusters
                sil = metrics.silhouette_score(GEMPA,
cluster.labels)
```

```
                silhouttes.append(sil)
                dbi = metrics.davies_bouldin_score(GEMPA,
cluster.labels)
                dbi_scores.append(dbi)
                ch = metrics.calinski_harabasz_score(GEMPA,
cluster.labels)
                ch_scores.append(ch)
            else:
                silhouttes.append(None)
                dbi_scores.append(None)
                ch_scores.append(None)
clusters
unique_values = []
for item in clusters:
    if item not in unique_values:
        unique_values.append(item)
print(unique_values)
tmp = 0
for i in range(len(ε_11)):
    for j in range(len(eps21)):
        for k in range(len(minpts)):
            tmp += 1
            cluster =
ST_DBSCAN(ε_1=ε_11[i],eps2=eps21[j],min_samples=minpts[k])
            cluster.fit(GEMPA)
            fig = plt.figure()
            ax = fig.add_subplot(projection='3d')
            ax.scatter3D(GEMPA['latitude'], GEMPA['longitude'],
GEMPA['time'],
            c=cluster.labels)
            if tmp < 9:
                plt.savefig('D:/AAA BISMILLAH TESIS/PERSIAPAN
UJIAN/GAMBAR PLOT1/IMG_0'+ str(tmp+1)+ '.png', dpi=(720))
                plt.close(fig)
            else:
                plt.savefig('D:/AAA BISMILLAH TESIS/PERSIAPAN
UJIAN/GAMBAR PLOT1/IMG_'+ str(tmp+1)+ '.png', dpi=(720))
                plt.close(fig)
##### PLOT SILHOUETTE
plt.figure(figsize=(12, 8))
for minpt in minpts:
    subset = hasil1[hasil1['MinPts'] == minpt]
```

```
    plt.plot(subset['E_1'], subset['Silhouette Coefficient'],
label=f'MinPts={minpt}')
plt.xlabel('E_1')
plt.ylabel('Silhouette Coefficient')
plt.title('Silhouette Coefficient vs E_1 for different MinPts
values')
plt.legend()
plt.grid(True)
# Simpan gambar hasil plot
plt.savefig('silhouette_coefficient_plot.png')
plt.show()
########### Plot DBI
plt.figure(figsize=(12, 8))
for minpt in minpts:
    subset = hasil1[hasil1['MinPts'] == minpt]
    plt.plot(subset['E_1'], subset['DBI], label=f'MinPts={minpt}')
plt.xlabel('E_1')
plt.ylabel('DBI')
plt.title('DBI vs E_1 for different MinPts values')
plt.legend()
plt.grid(True)
# Simpan gambar hasil plot
plt.savefig('DBI_plot.png')
plt.show()
########### Plot Calinski Harabahz
plt.figure(figsize=(12, 8))
for minpt in minpts:
    subset = hasil1[hasil1['MinPts'] == minpt]
    plt.plot(subset['E_1'], subset['Calinski Harabahz'],
label=f'MinPts={minpt}')
plt.xlabel('E_1')
plt.ylabel('Calinski Harabahz')
plt.title('Calinski Harabahz vs E_1 for different MinPts values')
plt.legend()
plt.grid(True)
# Simpan gambar hasil plot
plt.savefig('Calinski_Harabahz_plot.png')
plt.show()
########### PARAMETER TERBAIK
# Load the provided Excel file
file_path = 'D:/AAA BISMILLAH TESIS/PERSIAPAN UJIAN/Hasil Simulasi
Coba5.xlsx'
```

```
df = pd.read_excel(file_path)
# Display the first few rows of the dataframe to understand its
structure
print(df.head())
# Drop rows with NaN values
df_clean = df.dropna(subset=['Silhouette Coefficient', 'DBI',
'Calinski Harabahz']).copy()
# Normalisasi setiap metrik
SettingWithCopyWarning
df_clean.loc[:, 'Silhouette Norm'] = (df_clean['Silhouette
Coefficient'] - df_clean['Silhouette Coefficient'].min()) /
(df_clean['Silhouette Coefficient'].max() - df_clean['Silhouette
Coefficient'].min())
df_clean.loc[:, 'DBI Norm'] = (df_clean['DBI'].max() -
df_clean['DBI']) / (df_clean['DBI'].max() - df_clean['DBI'].min())
df_clean.loc[:, 'CHI Norm'] = (df_clean['Calinski Harabahz'] -
df_clean['Calinski Harabahz'].min()) / (df_clean['Calinski
Harabahz'].max() - df_clean['Calinski Harabahz'].min())
# Hitung rata-rata normalisasi dari ketiga metrik
df_clean.loc[:, 'Average Score'] = (df_clean['Silhouette Norm'] +
df_clean['DBI Norm'] + df_clean['CHI Norm']) / 3
# Temukan kombinasi parameter dengan rata-rata terbaik
best_params = df_clean.loc[df_clean['Average Score'].idxmax()]
print(best_params[['E_1', 'Eps2', 'MinPts', 'Silhouette Coefficient'
```