DAFTAR PUSTAKA

Award, M., & Khanna, R. (2015). *Efficient Machine Learning* (New York). Apress Media.

Babcock University, F.Y, O., J.E.T, A., O, A., J. O, H., O, O., & J, A. (2017). Supervised Machine Learning Algorithms: Classification and Comparison. *International Journal of Computer Trends and Technology*, *48*(3), 128–138. https://doi.org/10.14445/22312803/IJCTT-V48P126

Bi, Q., Goodman, K. E., Kaminsky, J., & Lessler, J. (2019). What is Machine Learning? A Primer for the Epidemiologist. *American Journal of Epidemiology*, kwz189. https://doi.org/10.1093/aje/kwz189

Dasgupta, A., & Nath, A. (2016). Classification of Machine Learning Algorithms. *International Journal of Innovative Resear Ch in Advanced Engineering*, *3*.

Febrianti, W. D., & Najah, N. K. J. (2024). *Analisis Klasifikasi Kepuasan Penumpang Maskapai Penerbangan Menggunakan Metode Support Vector Machine, Decision Tree, Dan Random Forest*. *20*(10). https://doi.org/10.12962/j27213862.vxix.xxxx

Fitri, E. N., Winarno, S., Budiman, F., Rohmani, A., Zeniarja, J., & Sugiarto, E. (2023). DECISION TREE SIMPLIFICATION THROUGH FEATURE SELECTION APPROACH IN SELECTING FISH FEED SELLERS. *Jurnal Teknik Informatika (Jutif)*, *4*(2), 301–309. https://doi.org/10.52436/1.jutif.2023.4.2.747

Gusril, H. (2016). *STUDI KUALITAS AIR MINUM PDAM DI KOTA DURI RIAU*. *8*(2), 190–196.

Huang, S., CAI, N., & PACHECO, P. P. (2018). Applications of Support Vector Machine (SVM) Learning in Cancer Genomics. *Cancer Genomics & Proteomics*, *15*(1). https://doi.org/10.21873/cgp.20063

Ibrahem Ahmed Osman, A., Najah Ahmed, A., Chow, M. F., Feng Huang, Y., & El-Shafie, A. (2021). Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia. *Ain Shams Engineering Journal*, *12*(2), 1545–1556. https://doi.org/10.1016/j.asej.2020.11.011

Karo, I. M. K. (2020). Implementasi Metode XGBoost dan Feature Importance untuk Klasifikasi pada Kebakaran Hutan dan Lahan. *Journal of Software Engineering*, *1*(1).

Lestari, S., Akmaludin, A., & Badrul, M. (2020). IMPLEMENTASI KLASIFIKASI NAIVE BAYES UNTUK PREDIKSI KELAYAKAN PEMBERIAN PINJAMAN PADA KOPERASI ANUGERAH BINTANG CEMERLANG. *PROSISKO: Jurnal Pengembangan Riset dan Observasi Sistem Komputer*, *7*(1), 8–16. https://doi.org/10.30656/prosisko.v7i1.2129

Mahesh, B. (2020). Machine Learning Algorithms—A Review. *International Journal of Science and Research (IJSR)*, *9*(1), 381–386. https://doi.org/10.21275/ART20203995

Ma'ruf, F. A., Adiwijaya, & Wisesty, U. N. (2019). Analysis of the influence of Minimum Redundancy Maximum Relevance as dimensionality reduction method on cancer classification based on microarray data using Support Vector Machine classifier. *Journal of Physics: Conference Series*, *1192*, 012011. https://doi.org/10.1088/1742-6596/1192/1/012011

Mowri, R. A., Siddula, M., & Roy, K. (2023). Is iterative feature selection technique efficient enough? A comparative performance analysis of RFECV feature selection technique in ransomware classification using SHAP. *Discover Internet of Things*, *3*(1), 21. https://doi.org/10.1007/s43926-023-00053-2

Pratama, A. R. I., Latipah, S. A., & Sari, B. N. (2022). OPTIMASI KLASIFIKASI CURAH HUJAN MENGGUNAKAN SUPPORT VECTOR MACHINE (SVM) DAN RECURSIVE FEATURE ELIMINATION (RFE). *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, *7*(2), 314–324. https://doi.org/10.29100/jipi.v7i2.2675

Pratama, I., Chandra, A. Y., & Presetyaningrum, P. T. (2022). Seleksi Fitur dan Penanganan Imbalanced Data menggunakan RFECV dan ADASYN. *Jurnal Eksplora Informatika*, *11*(1), 38–49. https://doi.org/10.30864/eksplora.v11i1.578

Shi, K., Shi, R., Fu, T., Lu, Z., & Zhang, J. (2024). A Novel Identification Approach Using RFECV–Optuna–XGBoost for Assessing Surrounding Rock Grade of Tunnel Boring Machine Based on Tunneling Parameters. *Applied Sciences*, *14*(6), 2347. https://doi.org/10.3390/app14062347

Sudianto, S. (2022). Analisis Kinerja Algoritma Machine Learning Untuk Klasifikasi Emosi. *Building of Informatics, Technology and Science (BITS)*, *4*(2). https://doi.org/10.47065/bits.v4i2.2261

Wandrivel, R., Suharti, N., & Lestari, Y. (2012). Kualitas Air Minum Yang Diproduksi Depot Air Minum Isi Ulang Di Kecamatan Bungus Padang Berdasarkan Persyaratan Mikrobiologi. *Jurnal Kesehatan Andalas*, *1*(3). https://doi.org/10.25077/jka.v1i3.84

Wang, Y., Pan, Z., Zheng, J., Qian, L., & Li, M. (2019). A hybrid ensemble method for pulsar candidate classification. *Astrophysics and Space Science*, *364*(8), 139. https://doi.org/10.1007/s10509-019-3602-4

WHO. (2023, September 13). *Drinking-water*. https://www.who.int/news-room/fact-sheets/detail/drinking-water

Wibawa, A. P., Purnama, M. G. A., Akbar, M. F., & Dwiyanto, F. A. (2018). *Metode-metode Klasifikasi*. *3*(1).

Yanti, C. A., & Akhri, I. J. (2021). PERBEDAAN UJI KORELASI PEARSON, SPEARMAN DAN KENDALL TAU DALAM MENGANALISIS KEJADIAN DIARE. *Jurnal Endurance*, *6*(1), Article 1. https://doi.org/10.22216/jen.v6i1.137

Yu, H., & Kim, S. (2012). SVM Tutorial—Classification, Regression and Ranking. Dalam G. Rozenberg, T. Bäck, & J. N. Kok (Ed.), *Handbook of Natural Computing* (hlm. 479–506). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-92910-9_15

Yuniar, M. N. (2023). *Klasifikasi Kualitas Air Bersih Menggunakan Metode Naïve baiyes*. *5*(1).

Zulaikhah Hariyanti Rukmana, S., Aziz, A., & Harianto, W. (2022). OPTIMASI ALGORITMA K-NEAREST NEIGHBOR (KNN) DENGAN NORMALISASI DAN SELEKSI FITUR UNTUK KLASIFIKASI PENYAKIT LIVER. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *6*(2), 439–445. https://doi.org/10.36040/jati.v6i2.4722

Lampiran 1 Contoh Dataset: Pengukuran air harian

Manajemen Kualitas Air untuk Tahun 2024 bulan Januari IPA Pandang-Pandang PDAM Gowa

Tanggal /hari	Waktu pengukuran (Kalau rata2, "R(kali)")	Kualitas air									
		Air baku				Air Bersih					
		Kekeruhan (NTU)	pH (-)	Suhu (°C)	Alkalinitas (ppm)	Kekeruhan (NTU)	pH (-)	Suhu (°C)	Klor bebas (mg/L)	Alkalinitas (ppm)	TDS (ppm)
1 (Senin)	-	-	-	-	-	-	-	-	-	-	-
2 (Selasa)	10	10,3	7,2	30,1	69,5	1,46	7,2	29,7	0,21	62,0	117
3 (Rabu)	7	10,1	7,2	35,0	67,0	1,32	7,3	30,5	-	58,0	116
4 (Kamis)	9	8,07	7,2	30,6	72,0	1,56	6,9	30,1	0,63	65,5	114
5 (Jumat)	4	9,28	7,1	30,4	66,5	1,46	7,2	30,5	0,60	61,5	113
6 (Sabtu)	9	83,8	7,5	29,7	50,5	1,15	7,2	29,2	0,64	46,0	95
7 (Minggu)	-	-	-	-	-	-	-	-	-	-	-
8 (Senin)	7	78,1	7,6	28,4	36,0	0,75	7,2	28,6	0,73	35,5	68
9 (Selasa)	2	87,2	7,5	28,7	46,5	3,64	7,1	28,4	0,28	39,5	75
10 (Rabu)	1	81,3	7,7	28,2	43,5	1,57	7,3	28,6	0,51	33,0	80
11 (Kamis)	-	-	7,3	29,8	48,0	-	7,1	29,2	0,20	45,0	87
12 (Jumat)	4	43,0	7,4	29,6	51,0	1,2	7,3	27,8	0,20	53,0	92
13 (Sabtu)	5	52,2	7,4	29,1	62,5	2,94	7,2	28,9	-	55,5	99
14 (Minggu)	-	-	-	-	-	-	-	-	-	-	-
15 (Senin)	9	93,9	7,5	27,7	53,0	1,94	7,3	28,5	-	44,0	72
16 (Selasa)	9	207	7,7	26,8	47,0	3,30	7,1	27,1	-	27,5	52
17 (Rabu)	4	208	7,7	26,7	42,0	3,12	7,3	26,8	-	32,5	50
18 (Kamis)	6	172	7,7	26,6	39,5	4,55	7,1	26,9	0,80	29,5	51
19 (Jumat)	10	134	7,6	28,1	44,0	4,6	7,0	28,3	0,54	31,0	60
20 (Sabtu)	5	319	7,9	27,0	35,5	3,24	7,1	27,5	0,50	25,5	46
21 (Minggu)	-	-	-	-	-	-	-	-	-	-	-
22 (Senin)	2	173	7,7	29,3	28,0	5,0	7,0	28,4	0,71	39,0	55
23 (Selasa)	10	178,6	7,4	27,1	35,0	3,84	7,3	27,5	0,23	34,5	54
24 (Rabu)	9	152,2	8,1	26,7	38,5	1,04	7,3	27,0	0,44	25,5	50
25 (Kamis)	7	144,7	8,1	27,7	35,5	1,21	7,3	27,7	0,25	32,5	50
26 (Jumat)	4	144,7	8,0	28,1	38,5	1,01	7,3	28,3	0,44	35,0	53
27 (Sabtu)	5	130,4	8,0	27,2	37,5	0,98	7,3	27,6	0,35	35,5	47
28 (Minggu)	-	-	-	-	-	-	-	-	-	-	-
29 (Senin)	10	125,6	8,1	28,6	41	1,76	7,3	28,3	0,46	32,5	55
30 (Selasa)	7	115,6	-	27,6	46	1,11	-	28,0	0,61	40,5	53
31 (Rabu)	10	136,8		28,2	38	1,15	7,3	28,0	0,25	32,0	53
Minimum	-	8,07	7,1	26,60	28,0	0,75	6,90	26,8	0,20	25,5	46
Maksimum	-	319,20	8,1	35,00	72,0	5,00	7,30	30,5	0,80	65,5	117
Rata-rata	-	115,96	7,6	28,58	46,6	2,20	7,20	28,4	0,46	40,4	71

Lampiran 2 Contoh dataset : Data Sampling Internal

NO.	NAMA PELANGGAN	ALAMAT	NO.SR	PARAMETER FISIK										PARAMETER KIMIA				PARAMETER MIKROBIOLOGI		JAM
				HASIL ANALISA					STANDAR					HASIL ANALISA	HASIL ANALISA	STANDAR	STANDAR	HASIL ANALISA	STANDAR	
				BAU	RASA	TDS (ppm)	NTU	SUHU (°C)	BAU	RASA	TDS (ppm)	NTU	SUHU (°C)	pH	SISA CHLOR (mg/L)	pH	SISA CHLOR (mg/L)	BAKTERI COLIFORM	BAKTERI COLIFORM	
4	Paisal	BTN. Taborong Permai B2/6	0710302484	TDK BAU	TDK BERASA	102	3,10	29,7	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,23	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	9,21
5	Mardiana	BTN.Inayah Mega E/2	0710200186	TDK BAU	TDK BERASA	84	1,70	30,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,25	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	#####
6	Rena Angraena	Mutiara Azzahra D 03/10	0710302125	TDK BAU	TDK BERASA	79	0,60	29,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,6	0,25	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	9,41
7	Kartini	BTN. Bajeng Permai T/5	0330200248	TDK BAU	TDK BERASA	95	2,00	30,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,1	0,53	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	9,02
8	Munawar	BTN. Bajeng Permai P/12	0330200230	TDK BAU	TDK BERASA	82	2,29	29,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,4	0,50	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,54
9	Muhtar Sijaya	BTN. Lestari B4/13	0320100132	TDK BAU	TDK BERASA	77	2,60	30,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,2	0,54	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,10
10	Agus A.Md	BTN. Lestari C3/7	0320100156	TDK BAU	TDK BERASA	80	2,33	28,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,41	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,18
11	H.Anwar	Jl. Poros Limbung	0310100068	TDK BAU	TDK BERASA	94	2,87	29,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,46	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,32
12	Syarifuddin	Kalase'rena	0910100001	TDK BAU	TDK BERASA	80	0,85	28,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,4	0,44	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,30
13	Hj.Sukriati Dg.Ti'no	Ballabarang	0910100149	TDK BAU	TDK BERASA	86	0,75	28,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,2	0,23	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,05
14	Mangngemba	Kalase'rena	0910100165	TDK BAU	TDK BERASA	88	1,18	27,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,4	0,39	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,18
15	Hamzah Dg.Lewa	Pammantoang	0910100248	TDK BAU	TDK BERASA	89	0,84	28,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,52	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,10
16	Dunniati	Sela	0910100234	TDK BAU	TDK BERASA	86	1,07	27,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,43	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,07
17	Kartini Am	Jl.Dg.Tata	0130701728	TDK BAU	TDK BERASA	108	0,10	30,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,6	0,30	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	#####
18	Hj.Fitriani	Jl.Pesona 1 No.4	0130701784	TDK BAU	TDK BERASA	106	0,60	30,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,1	0,20	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	#####
19	Hendi Wijaya	Graha Satelit A1	0130800128	TDK BAU	TDK BERASA	122	0,50	29,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,2	0,20	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	#####
20	Hj.Saodah Dg.Ngada	Duta Mas Pertiwi A.16	0130701005	TDK BAU	TDK BERASA	59	2,80	27,4	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,2	0,20	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	#####
21	Serda Rifani	Jl.Sultan Hasanuddin	0130701014	TDK BAU	TDK BERASA	50	3,80	26,9	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,2	0,20	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	9,59
22	Jaya H.Kasman	Taman Pesona Asri F No.1	0130701676	TDK BAU	TDK BERASA	56	3,90	27,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,1	0,20	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	#####
23	Mahfuddin	BTN. Mutiara Permai Blok S/	0120801614	TDK BAU	TDK BERASA	50	2,17	28,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,24	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,00
24	Abd. Murad	Jl. Malino Belakang Pasar	0120500701	TDK BAU	TDK BERASA	55	1,13	28,8	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,32	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,10
25	Didi	BTN. Yusuf Bauty Garden A1/	0120601332	TDK BAU	TDK BERASA	55	2,97	28,4	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,1	0,35	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,30
26	Lukman	BTN. Griya Persada B1/7	0120601278	TDK BAU	TDK BERASA	45	1,16	28,3	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,2	0,30	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	9,00
27	Sulaiman Dg.Ngawing	Jl. Poros Malino	0120501518	TDK BAU	TDK BERASA	53	0,87	28,2	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,26	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,22
28	Mustari	Jl. Malino Belakang Pasar	0120500700	TDK BAU	TDK BERASA	54	0,98	28,3	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,30	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,15
29	Fatimah Dg.Kenang	Jl.Poros Balang-Balang	0410100061	TDK BAU	TDK BERASA	47	3,40	28,8	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,7	0,38	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,30
30	(MBR 19) Susanti	Bontomanai	0430100442	TDK BAU	TDK BERASA	47	2,30	29,5	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,8	0,45	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,35
31	Abdi Karya Negara	Jalan STTP	0420100392	TDK BAU	TDK BERASA	47	2,70	29,4	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,0	0,43	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,15
32	(MBR 19) Fatmawati	Griya Riski Abadi E/3	0420100329	TDK BAU	TDK BERASA	45	2,40	29,9	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,8	0,31	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,17
33	(MBR 19) Yenni Sri Wahyuni	BTN.Bumi Batara Mawang B.	0440201048	TDK BAU	TDK BERASA	46	2,04	29,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,8	0,40	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,40
34	Dg.Haruddin	Lantebung	0450300005	TDK BAU	TDK BERASA	66	5,27	26,2	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,21	6.5-8.5	0.2-0.5	POSITIF E.COLI	NEGATIF E.COLI	8,33
35	Raminah Dg.Kanang	Lantebung	0450300008	TDK BAU	TDK BERASA	68	4,91	25,7	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	7,3	0,17	6.5-8.5	0.2-0.5	POSITIF E.COLI	NEGATIF E.COLI	8,25
36	(MBR) Mutiara Dg.Ngasseng	Sailong	0610102146	TDK BAU	TDK BERASA	49	1,14	26,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,6	0,23	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,31
37	(MBR) Sala	Sailong	0610102557	TDK BAU	TDK BERASA	56	1,56	26,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,5	0,21	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,36
38	(MBR) Ansar Dg.Sese	Lamuru	0610102561	TDK BAU	TDK BERASA	44	1,81	27,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,5	0,30	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,40
39	Isperianto	Lamuru	0610100344	TDK BAU	TDK BERASA	46	1,69	26,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,7	0,24	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	8,47
40	Dg.Gassing	Marannu 01	0610101529	TDK BAU	TDK BERASA	47	1,37	27,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,5	0,24	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	9,01
41	Herlina Dg.Kebo	Marannu	0610101700	TDK BAU	TDK BERASA	51	1,97	26,0	TDK BAU	TDK BERASA	300	3	Suhu Udarah ±	6,6	0,24	6.5-8.5	0.2-0.5	NEGATIF E.COLI	NEGATIF E.COLI	9,09

Lampiran 3 *Source Code Program* : Model SVM

```
import numpy as np
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score,
confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

class OptimizedSVMClassifier:
    def __init__(self, kernel='rbf', C= 1000, gamma='scale', max_iter=1000, tol=1e-5):
        self.kernel = kernel
        self.C = C
        self.gamma = gamma
        self.max_iter = max_iter
        self.tol = tol
        self.alpha = None
        self.b = None
        self.support_vectors = None
        self.support_vector_labels = None

    def rbf_kernel(self, X1, X2):
        if self.gamma == 'scale':
            # Hitung gamma berdasarkan data
            n_features = X1.shape[1]
            self.gamma = 1 / (n_features * X1.var())
        elif self.gamma == 'auto':
            # Alternatif lain jika 'auto' digunakan
            self.gamma = 1 / X1.shape[1]

        dist_matrix = np.sum(X1**2, axis=1).reshape(-1, 1) + np.sum(X2**2, axis=1) - 2
* np.dot(X1, X2.T)
        return np.exp(-self.gamma * dist_matrix)

    def train(self, X, y):
        X = X.to_numpy() if hasattr(X, 'to_numpy') else X
        y = np.where(y <= 0, -1, 1)
        n_samples, n_features = X.shape
        K = self.rbf_kernel(X, X)
        self.alpha = np.zeros(n_samples)
        self.b = 0

        def objective_function(i, j):
            return self.alpha[i] + self.alpha[j] - 0.5 * K[i,i] * self.alpha[i]**2 - \
                0.5 * K[j,j] * self.alpha[j]**2 - y[i] * y[j] * K[i,j] * self.alpha[i] *
self.alpha[j] - \
                y[i] * self.alpha[i] * self.b - y[j] * self.alpha[j] * self.b

        for _ in range(self.max_iter):
            num_changed_alphas = 0
            for i in range(n_samples):
                Ei = np.sum(self.alpha * y * K[:,i]) + self.b - y[i]
                if (y[i] * Ei < -self.tol and self.alpha[i] < self.C) or (y[i] * Ei > self.tol and
self.alpha[i] > 0):
```

```
                j = np.random.choice([x for x in range(n_samples) if x != i])
                Ej = np.sum(self.alpha * y * K[:,j]) + self.b - y[j]

                alpha_i_old, alpha_j_old = self.alpha[i], self.alpha[j]

                if y[i] != y[j]:
                    L = max(0, self.alpha[j] - self.alpha[i])
                    H = min(self.C, self.C + self.alpha[j] - self.alpha[i])
                else:
                    L = max(0, self.alpha[i] + self.alpha[j] - self.C)
                    H = min(self.C, self.alpha[i] + self.alpha[j])

                if L == H:
                    continue

                eta = 2 * K[i,j] - K[i,i] - K[j,j]
                if eta >= 0:
                    continue

                self.alpha[j] = alpha_j_old - y[j] * (Ei - Ej) / eta
                self.alpha[j] = max(L, min(H, self.alpha[j]))

                if abs(self.alpha[j] - alpha_j_old) < 1e-5:
                    continue

                self.alpha[i] = alpha_i_old + y[i] * y[j] * (alpha_j_old - self.alpha[j])

                b1 = self.b - Ei - y[i] * (self.alpha[i] - alpha_i_old) * K[i,i] - \
                     y[j] * (self.alpha[j] - alpha_j_old) * K[i,j]
                b2 = self.b - Ej - y[i] * (self.alpha[i] - alpha_i_old) * K[i,j] - \
                     y[j] * (self.alpha[j] - alpha_j_old) * K[j,j]

                if 0 < self.alpha[i] < self.C:
                    self.b = b1
                elif 0 < self.alpha[j] < self.C:
                    self.b = b2
                else:
                    self.b = (b1 + b2) / 2

                num_changed_alphas += 1

            if num_changed_alphas == 0:
                break

        sv = self.alpha > 1e-3
        self.support_vectors = X[sv]
        self.support_vector_labels = y[sv]
        self.alpha = self.alpha[sv]

    def predict(self, X):
        X = X.to_numpy() if hasattr(X, 'to_numpy') else X
        K = self.rbf_kernel(X, self.support_vectors)
        y_pred = np.sign(np.dot(K, self.alpha * self.support_vector_labels) + self.b)
```

```
        return np.where(y_pred == -1, 0, 1)

    def evaluate(self, X_test, y_test):
        y_pred = self.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred, average='weighted')
        precision = precision_score(y_test, y_pred, average='weighted')
        conf_matrix = confusion_matrix(y_test, y_pred)
        class_report = classification_report(y_test, y_pred)
        return accuracy, f1, recall, precision, conf_matrix, class_report

# Create and train the classifier
classifier = OptimizedSVMClassifier(C= 1000, gamma='scale')

print("Training optimized SVM classifier with C=1 and gamma='scale'...")
classifier.train(X_train, y_train)

# Evaluate the classifier on training data
accuracy_train, f1_train, recall_train, precision_train, conf_matrix_train,
class_report_train = classifier.evaluate(X_train, y_train)
print(f"Training Accuracy: {accuracy_train:.4f}")
print(f"Training F1 Score: {f1_train:.4f}")
print(f"Training Recall: {recall_train:.4f}")
print(f"Training Precision: {precision_train:.4f}")

print("\nTraining Classification Report")
print(class_report_train)

# Visualize confusion matrix for training data
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_train, annot=True, fmt='d', cmap='Blues', xticklabels=['Non-
Potable', 'Potable'], yticklabels=['Non-Potable', 'Potable'])
plt.title('Confusion Matrix - Training Data')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()

# Evaluate the classifier on test data
accuracy_test, f1_test, recall_test, precision_test, conf_matrix_test, class_report_test =
classifier.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy_test:.4f}")
print(f"Test F1 Score: {f1_test:.4f}")
print(f"Test Recall: {recall_test:.4f}")
print(f"Test Precision: {precision_test:.4f}")

print("\nTest Classification Report")
print(class_report_test)

# Visualize confusion matrix for test data
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_test, annot=True, fmt='d', cmap='Blues', xticklabels=['Non-
Potable', 'Potable'], yticklabels=['Non-Potable', 'Potable'])
```

```
plt.title('Confusion Matrix - Test Data')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```

Lampiran 4 Source Code Program : Model XGBoost

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, recall_score,
precision_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
from xgboost import XGBClassifier

# Set random seed for reproducibility
np.random.seed(42)

class OptimizedXGBoostClassifier:
    def __init__(self, n_estimators=100, learning_rate=0.1, max_depth=5,
random_state=42):
        self.model = XGBClassifier(
            n_estimators=n_estimators,
            learning_rate=learning_rate,
            max_depth=max_depth,
            random_state=random_state
        )

    def train(self, X, y):
        self.model.fit(X, y)

    def predict(self, X):
        return self.model.predict(X)

    def evaluate(self, X_test, y_test):
        y_pred = self.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred, average='weighted')
        precision = precision_score(y_test, y_pred, average='weighted')
        conf_matrix = confusion_matrix(y_test, y_pred)
        class_report = classification_report(y_test, y_pred)
        return accuracy, f1, recall, precision, conf_matrix, class_report

    def save_model(self, filename):
        joblib.dump(self.model, filename)

    @classmethod
    def load_model(cls, filename):
        loaded_model = joblib.load(filename)
        classifier = cls()
```

```
        classifier.model = loaded_model
        return classifier

# Assuming you have your data in a DataFrame called 'df'
# If not, load your data here
# df = pd.read_csv('your_data.csv')

# Split features and target
X = df.drop('Potability', axis=1)
y = df['Potability']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train the classifier
classifier = OptimizedXGBoostClassifier()

print("Training XGBoost classifier...")
classifier.train(X_train, y_train)

# Save the trained model
classifier.save_model('optimized_xgboost_model.joblib')

# Load the saved model for evaluation
loaded_classifier =
OptimizedXGBoostClassifier.load_model('optimized_xgboost_model.joblib')

# Function to evaluate and visualize results
def evaluate_and_visualize(classifier, X, y, dataset_name):
    accuracy, f1, recall, precision, conf_matrix, class_report =
classifier.evaluate(X, y)

    print(f"\n{dataset_name} Results:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"F1 Score: {f1:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"Precision: {precision:.4f}")

    print(f"\n{dataset_name} Classification Report:")
    print(class_report)

    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Non-Potable', 'Potable'],
                yticklabels=['Non-Potable', 'Potable'])
    plt.title(f'Confusion Matrix - {dataset_name}')
    plt.ylabel('True Label')
```

```
    plt.xlabel('Predicted Label')
    plt.show()

# Evaluate on training data
evaluate_and_visualize(loaded_classifier, X_train, y_train, "Training Data")

# Evaluate on test data
evaluate_and_visualize(loaded_classifier, X_test, y_test, "Test Data")
```

Lampiran 5 *Souce Code* Program : Seleksi Fitur RFECV

```
from sklearn.svm import SVC, LinearSVC
from sklearn.feature_selection import RFECV
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class FeatureSelector:
    def __init__(self, estimator, cv=5):
        self.estimator = estimator
        self.cv = cv
        self.rfecv = None

    def fit(self, X, y):
        self.rfecv = RFECV(
            estimator=self.estimator,
            step=1,
            cv=StratifiedKFold(self.cv),
            scoring='accuracy'
        )
        self.rfecv.fit(X, y)

    def transform(self, X):
        return self.rfecv.transform(X)

    def get_selected_features(self, feature_names):
        return feature_names[self.rfecv.support_]

    def plot_scores(self):
        plt.figure(figsize=(10, 5))
        plt.xlabel("Number of features")
        plt.ylabel("Cross-validation score")
        plt.plot(range(1, len(self.rfecv.cv_results_['mean_test_score']) + 1),
                 self.rfecv.cv_results_['mean_test_score'])
        plt.title('Feature Selection with RFECV')
        plt.show()

def train_model(X_train, y_train, kernel='rbf', C= 1000, gamma='scale'):
    model = SVC(kernel=kernel, C=C, gamma=gamma, random_state=42)
    model.fit(X_train, y_train)
    return model

def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
```

```
    print("\nModel Evaluation:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"*Recall*: {*recall*:.4f}")
    print(f"F1-score: {f1:.4f}")

    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

def main(X_train, y_train, X_test, y_test):
    # Feature Selection
    print("RFECV with Linear SVC for feature selection")
    selector = FeatureSelector(LinearSVC(C=1, random_state=42))
    selector.fit(X_train, y_train)

    selected_features = selector.get_selected_features(X_train.columns)
    X_train_selected = selector.transform(X_train)
    X_test_selected = selector.transform(X_test)

    print(f"Number of optimal features: {selector.rfecv.n_features_}")
    print(f"Selected features: {selected_features.tolist()}")
    print(f"Best score: {np.max(selector.rfecv.cv_results_['mean_test_score']):.2f}")

    selector.plot_scores()

    # Train and evaluate model
    final_model = train_model(X_train_selected, y_train)
    evaluate_model(final_model, X_test_selected, y_test)

if __name__ == "__main__":
    # Assuming X_train, y_train, X_test, y_test are already defined
    main(X_train, y_train, X_test, y_test)
```

Lampiran 6 *Source Code* Program : Seleksi fitur SelectKbest

```
from sklearn.svm import SVC
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class FeatureSelector:
    def __init__(self, k='all'):
        self.k = k
        self.selector = None

    def fit(self, X, y):
        self.selector = SelectKBest(score_func=f_classif, k=self.k)
        self.selector.fit(X, y)

    def transform(self, X):
        return self.selector.transform(X)

    def get_selected_features(self, feature_names):
        mask = self.selector.get_support()
        return feature_names[mask]

    def plot_scores(self, feature_names):
        plt.figure(figsize=(12, 6))
        scores = -np.log10(self.selector.pvalues_)
        plt.bar(range(len(scores)), scores)
        plt.xticks(range(len(scores)), feature_names, rotation='vertical')
        plt.xlabel("Features")
        plt.ylabel("-log10(p-value)")
        plt.title('Feature Importance Scores')
        plt.tight_layout()
        plt.show()

def train_model(X_train, y_train, kernel='rbf', C=1000, gamma='scale'):
    model = SVC(kernel=kernel, C=C, gamma=gamma, random_state=42)
    model.fit(X_train, y_train)
    return model

def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    print("\nModel Evaluation:")
    print(f"Accuracy: {accuracy:.4f}")
```

```
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-score: {f1:.4f}")

    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

def main(X_train, y_train, X_test, y_test):
    # Feature Selection
    print("SelectKBest with f_classif for feature selection")
    selector = FeatureSelector(k=3)  # Select top 3 features
    selector.fit(X_train, y_train)

    selected_features = selector.get_selected_features(X_train.columns)
    X_train_selected = selector.transform(X_train)
    X_test_selected = selector.transform(X_test)

    print(f"Selected features: {selected_features.tolist()}")

    selector.plot_scores(X_train.columns)

    # Train and evaluate model
    final_model = train_model(X_train_selected, y_train)
    evaluate_model(final_model, X_test_selected, y_test)

if __name__ == "__main__":
    # Assuming X_train, y_train, X_test, y_test are already defined
    main(X_train, y_train, X_test, y_test)
```