REFERENCES

1. Pang- Ning tan, Michael Steinbach, Vipin kumar, Introduction to Data Mining (2016), Always Learning, Pearson.
2. Gordon S. Linoff, Michael J. A. Berry, Data Mining Techniques for Marketing and customer relationship management (2011), Wiley.
3. Barry Leventhal, An introduction to Data Mining and other techniques for advanced analytics (2010), Journal of Direct, Data and Digital Marketing Practice, Palgrave Macmillan UK, Volume 12, Issue 2, PP 137-153.
4. Mehmed Kantardzic, Data Mining: Concepts, Models, Methods and Algorithms, (2011) John Wiley & Sons.
5. M. A. Deshmukh, R. A. Gulhane, Importance of Clustering in Data Mining (2016), International Journal of Scientific & Engineering Research, Volume 7, Issue 2, PP 247-251.
6. Chandan K. Reddy, Charu C. Aggarwal, Data Clustering Algorithms and Applications (2013), Chapman and Hall/CRC.
7. Mirkin, Boris, Core Data Analysis: Summarization (2019), Correlation and Visualization, Undergraduate Topics in Computer Science Series, Springer.
8. Chengqi Zhang Shichao Zhang Association Rule Mining (2002), Springer.
9. Modi, Krishna, Outlier Analysis Approaches in Data Mining (2016). International Journal of Innovative Research in Technology.
10. Jiawei Han, Michelin Kamber, Jian Pei, Data mining concepts and techniques (2012), Morgan Kaufmann.
11. Parneet KaurManpreet SinghGurpreet Singh Josan, Classification and prediction-based data mining algorithms to predict slow learners in education sector (2015), 3rd International Conference on Recent Trends in Computing, Procedia Computer Science, Volume 57, PP 500-508.
12. K Prasanna Jyothi, Dr R SivaRanjani, Dr Tusar Kanti Mishra, S Ranjan Mishra, A Study of Classification Techniques of Data Mining Techniques in Health-Related Research (2017), International Journal of Innovative Research in Computer and Communication Engineering, Volume 5, Issue 7, PP 13779-13786.

13. Amirah Mohamed Shahiri, Wahidah Husain, Nur‟aini Abdul Rashid, A Review on Predicting Student's Performance Using Data Mining Techniques (2015), Procedia Computer Science, Volume 72, Special Issue, PP 414-422.
14. Alaa F. Sheta, Sara Elsir M. Ahmed, Hossam Faris, A Comparison between Regression, Artificial Neural Networks and Support Vector Machines for Predicting Stock Market Index (2015), International Journal of Advanced Research in Artificial Intelligence, Volume 4, Issue 7, PP 55-63.
15. Festim Halili, Avni Rustemi, Predictive Modeling: Data Mining Regression Technique Applied in a Prototype (2016), International Journal of computer science and mobile computing, Volume 5, Issue 8, PP 207-215.
16. John O. Rawlings, Sastry G. Pantula, David A. Dickey, Applied Regression Analysis: A Research Tool (1998), Springer.
17. Philippe Esling, Carlos Agon. Time-series data mining. ACM Computing Surveys, Association for Computing Machinery, 2012, Volume 45, Issue 1 205.
18. Kenneth D. Lawrence, Stephan Kudyba, Ronald K. Klimberg, Data Mining Methods and Applications (2007), Auerbach Publications.
19. Velicer, Wayne & Fava, Joseph. Time Series Analysis. (2003).
20. Godwin, Harold & Okafor, Christian. Modified Trend and Seasonal Time Series Analysis for Operations: A Case Study of Soft Drink Production (2012). International Journal of Engineering Research in Africa. 7. 63-72.
21. Juang WC, Huang SJ, Huang FD, Cheng PW, Wann SR. Application of time series analysis in modelling and forecasting emergency department visits in a medical centre in Southern Taiwan. BMJ Open. 2017 Dec 1;7(11): e018628.
22. Machiwal, D., Jha, M.K. Methods for Time Series Analysis. In: Hydrologic Time Series Analysis: Theory and Practice. (2012). Springer, Dordrecht.
23. Shaltami, Osama & Bustany, Ilas. Water quality – A review (2021).
24. Dwivedi, Anil. Researches in Water Pollution: A Review. (2017).
25. Ahmed, Shahid and Ismail, Saba, Water Pollution and its Sources, Effects & Management: A Case Study of Delhi (2018), International Journal of Current Advanced Research, 07(2), pp. 10436-10442.
26. Yang, Y., Yan, B. & Shen, W. Assessment of point and nonpoint sources pollution in Songhua River Basin (2010), Northeast China by using revised water quality model. Chin. Geogr. Sci. 20, 30–36.

27. Dhote, Jayashree & Ingole, Sangita & Chavhan, Dr. Arvind. Review on Waste Water Treatment Technologies (2012). International Journal of Engineering Research and Technology.
28. Zia, Shaikh & Graham, D. & Dolfing, Wastewater Treatment: Biological. (2013).
29. Das Kangabam, R., Bhoominathan, S.D., Kanagaraj, S. et al. Development of a water quality index (WQI) for the Loktak Lake in India (2017). Appl Water Sci 7, 2907–2918
30. Md. Galal Uddin, Stephen Nash, Agnieszka I. Olbert, A review of water quality index models and their use for assessing surface water quality (2021), Ecological Indicators, Volume 122, 107218.
31. Howladar, M.F., Al Numanbakth, M.A. & Faruque, M.O. An application of Water Quality Index (WQI) and multivariate statistics to evaluate the water quality around Maddhapara Granite Mining Industrial Area (2018), Dinajpur, Bangladesh. Environ Syst Res 6, 13.
32. Adelagun, Ruth & Etim, Emmanuel & Godwin, Oko Emmanuel. Application of Water Quality Index for the Assessment of Water from Different Sources in Nigeria (2021).
33. Algalil, Fahd Abd, Aldhyani, Theyazn H. H, Al-Yaari, Mohammed, Alkahtani, Hasan, Maashi, Mashael, Water Quality Prediction Using Artificial Intelligence Algorithms, 2020 6659314.
34. Tiyasha, Tung, T.M. & Yaseen, Z.M. Deep Learning for Prediction of Water Quality Index Classification: Tropical Catchment Environmental Assessment. (2021). Nat Resour Res 30, 4235–4254
35. Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions (2021). SN COMPUT. SCI. 2, 420.
36. Bank, Dor & Koenigstein, Noam & Giryes, Raja. Autoencoders. (2020).
37. Zhang, Yue & Liu, Fangai. An Improved Deep Belief Network Prediction Model Based on Knowledge Transfer. (2020). Future Internet. 12. 188. 10.3390/fi12110188.
38. Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions (2021). J Big Data 8, 53.
39. Alex Sherstinsky, Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network, Physica D: Nonlinear Phenomena, 2020, Volume 404, 132306.
40. Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, Michael Weyrich, A survey on long short-term memory networks for time series prediction, 2021, Procedia CIRP, Volume 99, pp 650-655.

41. Shen, Guizhu & Tan, Qingping & Haoyu, Zhang & Zeng, Ping & Xu, Jianjun. Deep Learning with Gated Recurrent Unit Networks for Financial Sequence Predictions. (2018). Procedia Computer Science. 131. 895-903.
42. Fang, Yong & Yang, Shaoshuai & Zhao, Bin & Huang, Cheng. Cyberbullying Detection in Social Networks Using Bi-GRU with Self-Attention Mechanism. (2021). Information. 12. 171. 10.3390/info1204017.
43. Bryan Lim, Sercan O. Arik, Nicolas Loeff, Tomas Pfister, Temporal Fusion Transformers for interpretable multi-horizon time series forecasting (2019).
44. Hosna, A., Merry, E., Gyalmo, J. et al. Transfer learning: a friendly introduction (2022). J Big Data 9, 102.
45. Day, O., Khoshgoftaar, T.M. A survey on heterogeneous transfer learning (2017). J Big Data 4, 29
46. Mignone, Paolo & Pio, Gianvito & Džeroski, Sašo & Ceci, Michelangelo. Multi-task learning for the simultaneous reconstruction of the human and mouse gene regulatory networks. (2020). Scientific Reports. 10. 22295. 10.1038/s41598-020-78033-7.
47. Wu, Qingyao, et al. Online Transfer Learning with Multiple Homogeneous or Heterogeneous Source,s 2017. IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 7, pp. 1494–507. IEEE Xplore.
48. Gupta, Jaya, et al. Deep Learning (CNN) and Transfer Learning: A Review. Journal of Physics: Conference Series, 2022. vol. 2273, no. 1, p. 012029.
49. Shrestha, AK & Basnet, N., The Correlation and regression analysis of physicochemical parameters of River water for the evaluation of percentage contribution to electrical conductivity, 2018, Journal of Chemistry, pp. 1-9.
50. Mohammad Mirzavand & Reza Ghazavi, A Stochastic Modelling Technique for Groundwater Level Forecasting in an Arid Environment Using Time Series Methods, 2014, Water Resour Manage 29: 1315-1328.
51. Xuedi Zhang, Hui Qian, Jie Chen and Liang Qiao, Assessment of Groundwater Chemistry and Status in a Heavily Used Semi-Arid Region with Multivariate Statistical Analysis, 2014, Water 6(8):2212-2232.
52. Semko Rashid, Milad Mohammadan, koorosh Azizi, Prediction of Groundwater level Fluctuation using ANN and ANFIS in Lailakh Plain, 2015, Journal of renewable Natural Resource Bhutan 3.1, pp: 77-84.

53. Batool A., Samad N., Kazmi S. S., Ghufran M. A., Imad S., Shafqat M. and Mahmood T.Spring water quality and human health: an assessment of natural springs of margalla hills 2018. Islamabad zone- III. Int. J. Hydrol., 2(1): 41–46.
54. Sakaa, B., Brahmia, N., Chaffai, H. and Hani, A. Assessment of water quality index in an unmonitored river basin using multilayer perceptron neural networks and principal component analysis. Desalin. 2020. Water Treat., 200: 42-54.
55. Setshedi, K. J., Mutingwende, N. and Ngqwala, N. P. The use of artificial neural networks to predict the physicochemical characteristics of water quality in three district municipalities, Eastern Cape Province, 2021.South Africa. Int. J. Environ. Res. Public Health, 18(10): 5248-56.
56. Singh, B., Sihag, P., Singh, V. P., Sepahvand, A. and Singh, K. Soft computing technique-based prediction of water quality index. 2021.Water Supply, 21(8): 4015-4029.
57. Kulisz, M., Kujawska, J., Przysucha, B. and Cel, W. Forecasting water quality index in groundwater using artificial neural network. 2021. Energies,14(18):58-75.
58. A. Solanki, H. Aggarwal, and K. Khare, Predictive Analysis of Water Quality Parameters using Deep Learning, 2015. International Journal of Computer Applications, vol. 125, no. 9, pp. 0975-8887.
59. U. Ahmed, R. Mumtaz, H. Anwar, A.A. Shah, R. Irfan, J. Garca-Nieto, Efficient water quality prediction using supervised machine learning, (2019) Water 11, 2210.
60. C.V. Sillberg, P. Kullavanijaya, O. Chavalparit, Water quality classification by integration of attribute realization and support vector machine for the chao phraya river, Journal of Ecological Engineering 22 (2021), 70–86.
61. Wang, Y., Zhou, J., Chen, K., Wang, Y., & Liu, L. (2017). Water quality prediction method based on LSTM neural network. In 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE) (pp. 1-5). IEEE.
62. Zhenbo Li, Fang Peng, Bingshan Niu, Guangyao Li, Jing Wu, Zheng Miao, Water Quality Prediction Model Combining Sparse Auto-encoder and LSTM Network, IFAC-PapersOnLine, Volume 51, Issue 17, 2018, Pages 831-836.
63. Xu J, Wang K, Lin C, Xiao L, Huang X, Zhang Y. FM-GRU: A Time Series Prediction Method for Water Quality Based on seq2seq Framework. Water. 2021; 13(8):1031.
64. M. Yilma, Z. Kiflie, A. Windsperger, N. Gessese, Application of artificial neural network in water quality index prediction: a case study in little Akaki River, Addis Ababa, Ethiopia, Modeling Earth Systems and Environment 4 (2018), 175-187.

65. Srivastava, A., Cano, A. Analysis and forecasting of rivers pH level using deep learning (2022). Prog Artif Intell 11, 181–191.
66. G Tan, J Yan, C Gao, and S Yang, Prediction of water quality time series data based on least squares support vector machine, Procedia Engineering, Vol. 31, 2012, pp. 1194-1199.
67. WC Leong, A Bahadori, J Zhang, and Z Ahmad, Prediction of water quality index (WQI) using support vector machine (SVM) and least square-support vector machine (LS-SVM), International Journal of River Basin Management, Vol. 19, 2021, pp. 149-156.
68. Aldhyani, T. H. H., Al-Yaari, M., Alkahtani, H. & Maashi, M. Water quality prediction using artificial intelligence algorithms. Applied Bionics and Biomechanics 2020, 6659314 (2020).
69. Yang, Y. et al. A study on water quality prediction by a hybrid CNN-LSTM model with attention mechanism. Environmental Science and Pollution Research (2021) .
70. Kamaraj M, Rangarajan S (2022) Predicting the future land use and land cover changes for Bhavani basin, Tamil Nadu, India, using QGIS MOLUSCE plugin. Environ Sci Pollut Res (2022).
71. Arunkumar R, Thambusamy, Velmurugan (2021) An exploratory data analysis process on groundwater quality data. 54:41–48.
72. Haghiabi, Amir Hamzeh, et al. 'Water Quality Prediction Using Machine Learning Methods'. Water Quality Research Journal, vol. 53, no. 1, Feb. 2018, pp. 3–13.
73. Ubah, J.I., Orakwe, L.C., Ogbu, K.N. et al. Forecasting water quality parameters using artificial neural network for irrigation purposes. Sci Rep 11, 24438 (2021).
74. Kargar, K. et al. Estimating longitudinal dispersion coefficient in natural streams using empirical models and machine learning algorithms. Eng. Appl. Comput. Fluid Mech. 14, 311–322 (2020).
75. Alizadeh, M. J. et al. Effect of river flow on the quality of estuarine and coastal waters using machine learning models. Eng. Appl. Comput. Fluid Mech. 12(1), 810–823 (2018).
76. WHO 2019 Drinking-water https://www.who.int/news-room/factsheets/detail/drinking-water.
77. Basant, N., Gupta, S., Malik, A., Singh, K.P., 2010. Linear and nonlinear modelling for simultaneous prediction of dissolved oxygen and biochemical oxygen demand of the surface water -A case study. Chemometr.Intellig.Lab.Syst.104,172–180.
78. Heddam, S., Kisi, O., 2018. Modelling daily dissolved oxygen concentration using least square support vector machine, multivariate adaptive regression splines and M5 model tree. J. Hydrol. 559, 499–509.

79. Li, G., 2006. Stream temperature and dissolved oxygen modelling in the Lower Flint River Basin. PhD Dissertation. University of Georgia, Athens, GA.
80. Wang, Y., Zhou, J., Chen, K., Wang, Y., & Liu, L. (2017). Water quality prediction method based on LSTM neural network. In 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE) (pp. 1-5). IEEE.
81. Li L, Jiang P, Xu H, Lin G, Guo D, Wu H (2019) Water quality prediction based on recurrent neural network and improved evidence theory: a case study of Qiantang River, China. Environ Sci Pollut Res 26(19): 19879–19896.
82. Li L, Jiang P, Xu H, Lin G, Guo D, Wu H (2019) Water quality prediction based on recurrent neural network and improved evidence theory: a case study of Qiantang River, China. Environ Sci Pollut Res 26(19): 19879–19896
83. Liu J, Yu C, Hu Z, Zhao Y, Bai Y, Xie M, Luo J (2020) Accurate prediction scheme of water quality in smart mariculture with deep Bi-S-SRU learning network. IEEE Access 8:24784–24798
84. Yahya A, Saeed A, Ahmed AN, Binti Othman F, Ibrahim RK, Afan HA, Elshafie A (2019) Water quality prediction model-based support vector machine model for ungauged river catchment under dual scenarios. Water 11(6):1231.
85. L. Hu, C. Zhang, C. Hu, and G. Jiang, Use of grey system for assessment of drinking water quality: a case study of Jiaozuo city, China, Advances in Grey Systems Research, Springer Berlin Heidelberg, pp. 469-478, 2010
86. M. A. Tirabassi, A statistically based mathematical water quality model for a non-estuarine river system1 JAWRA Journal of the American Water Resources Association, Vol. 7, pp. 1221-1237, December 1971.
87. H. Liao and W. Sun. Forecasting and evaluating water quality of Chao Lake based on an improved decision tree method. Procedia Environmental Sciences 2 (2010): 970-979
88. M. Valdivia, D.W. Graham, and D. Werner, Climatic, Geographic and Operational Determinants of Trihalomethanes (THMs) in Drinking Water Systems Scientific reports, Vol. 6, 2016.
89. M Tarique, H. Khaleeq, and A. A. ElNour, A Reliable Wireless System for Water Quality Monitoring, Vo. 8, No. 3, 2016.
90. Y. Khan and C. S. See, Predicting and Analyzing Water Quality using Machine Learning: A Comprehensive Model, IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2016.

91. X. Li and J. Song, A New ANN-Markov Chain Methodology for Water Quality Prediction, International Joint Conference on Neural Networks, pp. 12-17 July 2015
92. A. Solanki, H. Aggarwal, & K. Khare, Predictive Analysis of Water Quality Parameters using Deep Learning, International Journal of Computer Applications, vol. 125, no. 9, pp. 0975-8887, Access from Google Scholar, Sept. 2015.
93. Jayti Bhatt, Jignesh Patoliya, Iot Based Water Quality Monitoring System, IRFIC, 21feb,2016.
94. Dong He, Li-Xin Zhang, "The Water Quality Monitoring System Based on WSN,"Institute of Mechanical and electronic information, China University of Geosciences (WuHan), WuHan,China, pp. 3661-3664, 2012.
95. Herzog, S., Worg ¨ otter, ¨ F., Parlitz, U., 2018. Data-Driven Modeling and Prediction of Complex Spatio-Temporal Dynamics in Excitable Media. Front. Appl. Math. Stat. 4
96. Tung Tiyasha, T.M., Yaseen, Z.M., 2020. A survey on river water quality modelling using artificial intelligence models: 2000–2020. J. Hydrol. 585, 124670.
97. Kannan V, Ramesh R, Sasikumar C. 2005. Study on groundwater characteristics and the effects of discharged effluents from textile units at Karur district. Journal of Environmental Biology, 26(2):269-272.
98. Zhang X, Qian H, Chen J, Qiao L. 2014. Assessment of Groundwater Chemistry and Status in a Heavily Used Semi-Arid Region with Multivariate Statistical Analysis. Water. 6(8):2212-2232.
99. Ubah, J.I., Orakwe, L.C., Ogbu, K.N. et al. Forecasting water quality parameters using artificial neural network for irrigation purposes. Sci Rep 11, 24438 (2021).
100. Haghiabi, Amir Hamzeh, et al. 'Water Quality Prediction Using Machine Learning Methods'. Water Quality Research Journal, vol. 53, no. 1, Feb. 2018, pp. 3–13.
101. Krishan G, Singh S, Kumar CP, Gurjar S, Ghosh NC. 2016. Assessment of Water Quality Index (WQI) of Groundwater in Rajkot District, Gujarat, India. Journal of Earth Science & Climatic Change. 7(3):1000341.

LIST OF PUBLICATIONS

Papers Published in International Conference Proceedings

1. Jitha P Nair, and M. S. Vijaya. Predictive Models for River Water Quality Using Machine Learning and Big Data Techniques - A Survey. 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), IEEE, 2021, pp. 1747-53. **(SCOPUS INDEXED)**
2. Jitha P Nair, M S Vijaya. Exploratory Data Analysis of Bhavani River Water Quality Index Data. In: Kumar, S., Hiranwal, S., Purohit, S.D., Prasad, M. (eds) Proceedings of International Conference on Communication and Computational Technologies. Algorithms for Intelligent Systems. Springer, Singapore. **(WEB OF SCIENCE INDEXED)**
3. Jitha P Nair, and M. S. Vijaya. River Water Quality Prediction and Index Classification Using Machine Learning. Journal of Physics: Conference Series, vol. 2325, no. 1, Aug. 2022, p. 012011. **(SCOPUS INDEXED)**

Papers Published in International Journals

1. Jitha P Nair, and M. S. Vijaya. Analysing and Modelling Dissolved Oxygen Concentration using Deep Learning Algorithm, International Journal of Mechanical Engineering, Volume No.7, pp. 12-22. **(SCOPUS INDEXED)**
2. Jitha P Nair, and Vijaya, M. S. (2023). Enhanced Water Quality Prediction Model with Seasonal Time Series Data. Journal of Data Acquisition and Processing, 38(1), 1283. (**SCOPUS INDEXED)**
3. Jitha P Nair, and Vijaya, M. S. (2023) Design and development of efficient water quality prediction models using variants of recurrent neural networks. European Chemical Bulletin. Vol.No.12 Special issue 5. **(SCOPUS INDEXED)**
4. Jitha P Nair, and Vijaya, M. S. (2023) Temporal Fusion Transformer: A Deep Learning Approach for Modelling and Forecasting River Water Quality Index. Nair. International Journal of Intelligent Systems and Applications in Engineering, 11(10s), 277-293. **(SCOPUS INDEXED)**

Papers in Review - International Journals

1. TFT Architecture and RNN Variants for Water Quality Prediction of Bharathapuzha River - International Journal of Information Technology
2. Transfer Learning for Improved Deep Neural Network Models in River Water Quality Index Prediction -Journal of Network and Computer Application.
3. Transfer Learning and Temporal Fusion Transformer for Enhanced River Water Quality Index Prediction - Expert System with Application.
4. Heterogeneous Transfer Learning for Modelling and Forecasting River Water Quality Index - Journal of Cloud Computing

APPENDIX A- DATASETS

WQI-PCA Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1	Temp	pH	Conduct	Turbidity	Phenolpt	Total Alk	Choloride	COD	TKN	Ammoni	Hardnes	CaHardr	MgHard	Sulphate	Sodium	TSS	TDS	FDS	Phosphs	Boron	Potassi	BOD	Fluoride	Nitrate-N	TC	FC	Station	Latitude	Longitud	Year	DO	WQI
2	25	7.95	340	2	0	111	21	4	0.1	0.25	118	74	44	12	27.1	300	190	174	0.11	0.1	2.67	0.89	0.12	1.1	88	80	1	76.5911	11.0931	2016	6.99	76.79
3	24	7.46	339	2	0	110	21	3.9	0.1	0.25	118	74	44	12.5	27.1	300	190	174	0.11	0.1	2.67	0.87	0.18	1.1	98	80	1	76.5911	11.0931	2016	6.97	77.1
4	25	7.5	339	2	0	112	22	4	0.09	0.25	119	74.5	44	12	27.2	300	189	174	0.11	0.1	2.66	0.89	0.18	1.1	118	80	1	76.5911	11.0931	2016	6.81	77.12
5	25	7.18	340	2	0	111	21	3.9	0.1	0.25	119	74.5	43.5	12	27.2	300	189	174.5	0.11	0.1	2.66	0.88	0.18	1	86	79.5	1	76.5911	11.0931	2016	7.19	72.45
6	25	7.45	340	2	0	110	20	4	0.1	0.25	119	74	43.5	12.5	27	300	189	174.5	0.11	0.1	2.67	0.85	0.18	1.2	65	79.5	1	76.5911	11.0931	2016	7.3	81.9
7	24	7.05	342	2	0	110	20	4	0.09	0.25	119	73.5	43.5	12.5	27.1	300	190	174	0.11	0.1	2.67	0.87	0.17	1	105	79	1	76.5911	11.0931	2016	7.39	72.47
8	24	7.4	341	2	0	112	20	4	0.1	0.25	118	73.5	44	12	27.1	300	189	174	0.11	0.1	2.66	0.82	0.17	1.2	83	79.5	1	76.5911	11.0931	2016	7.06	81.7
9	25	7.38	339	2	0	111	21	3.9	0.1	0.25	118	73.5	44	12	27	300	190	174	0.11	0.1	2.66	0.81	0.17	1.2	113	80	1	76.5911	11.0931	2016	7.02	81.72
10	25	7.56	340	2	0	112	21	3.9	0.1	0.25	118	74	44	12.5	27.1	300	189	173.5	0.11	0.1	2.66	0.88	0.18	1.2	65	80	1	76.5911	11.0931	2016	6.97	81.81
11	25	7.1	340	2	0	111	21	4	0.11	0.25	117.5	74	44	12	27.1	300	188	173	0.11	0.1	2.66	0.82	0.18	1.2	85	80	1	76.5911	11.0931	2016	7.39	81.86
12	24	7.27	339	2	0	112	22	4	0.11	0.25	117	74	44.5	12	27.1	300	189	173	0.11	0.1	2.68	0.89	0.18	1.1	88	80	1	76.5911	11.0931	2016	7.12	77.17
13	25	7.28	341	2	0	111	22	3.9	0.1	0.25	117.5	74.5	44.5	12	27	300	189	173.5	0.11	0.1	2.68	0.81	0.18	1.1	93	79.5	1	76.5911	11.0931	2016	7.38	77.18
14	24	7.3	339	2	0	110	22	3.9	0.1	0.25	117	74.5	44.5	11.5	27.1	300	189	173.5	0.11	0.1	2.67	0.83	0.19	1.2	69	79.5	1	76.5911	11.0931	2016	7.12	81.81
15	25	7.56	340	2	0	111	22	4	0.1	0.25	118	74.5	44	11.5	27	300	190	174	0.11	0.1	2.67	0.86	0.19	1.1	108	79.5	1	76.5911	11.0931	2016	7.27	77.33
16	25	7.06	341	2	0	112	20	4.1	0.1	0.25	118	75	44	11.5	27.1	300	190	174	0.11	0.1	2.68	0.89	0.19	1.1	83	79.5	1	76.5911	11.0931	2016	6.98	77.15
17	24	7.21	341	2	0	112	21	4	0.1	0.25	119	74.5	44	11.5	27.1	300	190	174.5	0.11	0.1	2.68	0.84	0.19	1.2	64	80	1	76.5911	11.0931	2016	7.3	81.89
18	25	7.2	340	2	0	110	21	4.1	0.11	0.25	119	74	43.5	12	27.1	300	190	174.5	0.11	0.1	2.67	0.87	0.18	1	91	80.5	1	76.5911	11.0931	2016	7.11	72.44
19	25	7.51	341	2	0	111	21	4.1	0.11	0.25	119	74	44	12	27	300	190	173.5	0.11	0.1	2.66	0.86	0.18	1.1	108	80.5	1	76.5911	11.0931	2016	7.04	77.18
20	25	7.35	341	2	0	111	20	4.1	0.1	0.25	118	73.5	43.5	12	27	300	191	174	0.11	0.1	2.66	0.87	0.18	1	63	80	1	76.5911	11.0931	2016	6.98	72.37
21	24	7.37	339	2	0	110	20	4.1	0.1	0.25	117	73	43.5	12	27.1	300	191	174.5	0.11	0.1	2.66	0.88	0.18	1	117	80.5	1	76.5911	11.0931	2016	6.97	72.44
22	24	7.2	340	2	0	111	20	3.9	0.09	0.25	117	73	44	12	27.1	300	191	174	0.11	0.1	2.67	0.84	0.17	1.1	103	80.5	1	76.5911	11.0931	2016	6.97	76.38
23	25	7.01	339	2	0	110	20	3.9	0.09	0.25	117	74	44	12	27.2	300	191	174	0.11	0.1	2.68	0.89	0.17	1.2	101	80.5	1	76.5911	11.0931	2016	7.28	81.86
24	25	7.58	340	2	0	110	21	4.1	0.09	0.25	118	74	44.5	12.5	27.1	300	191	174	0.11	0.1	2.68	0.83	0.18	1.1	79	79.5	1	76.5911	11.0931	2016	6.8	77.02
25	25	7.14	340	2	0	111	21	4.1	0.09	0.25	117.5	74.5	44.5	12.5	27.2	300	191	174.5	0.11	0.1	2.68	0.88	0.18	1.2	107	80	1	76.5911	11.0931	2016	6.84	81.77
26	24	7.55	339	2	0	112	21	3.9	0.1	0.25	118	74.5	44.5	12	27.2	300	190	174.5	0.11	0.1	2.67	0.83	0.19	0.9	76	80.5	1	76.5911	11.0931	2016	7.33	67.85
27	25	7.06	341	2	0	112	21	4.1	0.1	0.25	119	74.5	44.5	12	27.1	300	190	174.5	0.11	0.1	2.68	0.87	0.18	1.1	112	80.5	1	76.5911	11.0931	2016	6.86	77.06
28	24	7.53	341	2	0	110	22	4	0.1	0.25	118.5	73.5	44	12	27.1	300	191	174	0.11	0.1	2.68	0.87	0.19	1.1	90	80.5	1	76.5911	11.0931	2016	7.03	77.2
29	25	7.49	340	2	0	112	21	4	0.11	0.25	118	73	44	12	27.1	300	191	174	0.11	0.1	2.67	0.9	0.19	0.9	81	80.5	1	76.5911	11.0931	2016	6.91	67.77
30	24	7.26	339	2	0	111	22	4.1	0.1	0.25	117	74	44	11.5	27.2	300	191	174	0.11	0.1	2.67	0.83	0.18	1	110	80.5	1	76.5911	11.0931	2016	6.93	72.33
31	24	7.58	339	2	0	111	21	4	0.11	0.25	117	74	43.5	11.5	27.1	300	191	174	0.11	0.1	2.67	0.83	0.17	1.2	109	80	1	76.5911	11.0931	2016	6.94	81.73
32	24	7.39	340	2	0	111	22	4	0.1	0.25	118	74	44	12	27.1	300	190	174	0.11	0.1	2.67	0.83	0.17	1.1	120	80	1	76.5911	11.0931	2016	7.37	77.19
33	24	7.29	335	2	0	113	23	4	0.1	0.25	120	72	48	12	28	300	190	170	0.11	0.1	2.68	0.81	0.18	0.9	109	80	1	76.5911	11.0931	2016	6.89	67.64
34	24	7.74	335	2	0	113	23	4	0.1	0.25	122	72	48	12	28	300	190	170	0.11	0.1	2.68	0.88	0.18	0.9	115	76	1	76.5911	11.0931	2016	7.02	67.84
35	24	7.76	330	2	0	115	23	4	0.1	0.25	124	72	50	12	28	300	185	170	0.11	0.1	2.68	0.84	0.19	0.9	83	75	1	76.5911	11.0931	2016	6.12	67.45
36	24	7.47	330	2	0	115	23	4	0.1	0.25	125	70	50	12.5	28.5	300	185	168	0.11	0.1	2.7	0.82	0.19	0.9	92	70	1	76.5911	11.0931	2016	6.51	67.54
37	26	7.64	329	2	0	118	25	4	0.1	0.25	125	70	50	12.5	28.5	300	185	168	0.11	0.1	2.7	0.85	0.18	0.8	80	70	1	76.5911	11.0931	2016	6.17	62.73
38	25	7.45	325	2	0	118	24	4.2	0.1	0.25	125	70	54	12.5	28.5	300	183	168	0.11	0.1	2.8	0.9	0.18	0.8	93	65	1	76.5911	11.0931	2016	6.59	62.99
39	25	7.42	325	2	0	118	25	4.1	0.1	0.25	125	70	54	12	28.5	300	183	168	0.11	0.1	2.8	0.8	0.18	0.8	107	65	1	76.5911	11.0931	2016	5.94	62.59
40	24	7.54	323	2	0	118	25	3.9	0.11	0.25	127	68	54	12	29	300	183	165	0.11	0.1	2.9	0.85	0.18	0.8	106	65	1	76.5911	11.0931	2016	5.99	62.85
41	24	7.18	320	2	0	119	25	3.9	0.11	0.25	127	69	58	12	29	300	183	165	0.11	0.1	2.9	0.83	0.17	0.8	105	60	1	76.5911	11.0931	2016	6.29	62.67
42	26	7.43	318	2	0	119	26	4	0.1	0.25	127	69	58	13	29	300	180	165	0.11	0.1	2.9	0.87	0.17	0.7	105	60	1	76.5911	11.0931	2016	6.03	57.99

WQI-SA Dataset

WQI-BP Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN
1	Temp	pH	Conducti	Turbidity	Phenolpl	Total Alk	Chloride	COD	TKN	Ammonia	Hardness	Ca.Hardr	Mg.Hardr	Sulphate	Sodium	TDS	FDS	TSS	Phosphat	Boron	Pottassiu	BOD	Fluoride	Nitrate-N	TC	FC	Dew	Humidity	Sealevel	Precipita	Precipcov	Windspee	Winddir	Cloudcov	Visibility	Latitude	Longitud	Date	DO	WQI
2	29	7.3055	312	2	0	77	45	4	0.1	0.25	105	54	53	0.09901	16	180	99	300	0.16	0.1	6.37768	1.01485	0.57677	1.28997	320	281	15.7	59.3	1016.6	0	0	16.3	52.9	27.4	5.5	10.5026N	76.0120E	01-01-2020	6.12138	95.6217
3	27	7.59904	316	2	0	75	43	4	0.1	0.25	100	50	51	0.30502	15	173	101	300	0.16	0.1	6.40487	1.02723	0.45276	1.0332	306	272	14.6	56.72	1017.1	0	0	14.4	62.3	17.9	6	10.5026N	76.0120E	02-01-2020	6.24691	82.8914
4	30	7.43013	295	2	0	76	39	4	0.09	0.25	104	51	50	0.13964	17	171	94	300	0.16	0.1	6.46414	1.02239	0.50389	1.51296	338	258	13.4	51.89	1015.8	0	0	13.1	61.7	5.5	5.7	10.5026N	76.0120E	03-01-2020	6.11927	105.654
5	28	7.27194	290	2	0	79	38	4	0.1	0.25	108	53	50	0.28487	17	175	102	300	0.16	0.1	6.42038	1.01685	0.45001	1.18582	312	264	13.6	53.06	1015.7	0	0	15.4	68.2	14.1	5.9	10.5026N	76.0120E	04-01-2020	6.25349	89.9749
6	30	7.63378	316	2	0	80	39	4	0.1	0.25	103	50	52	0.30761	18	180	101	300	0.16	0.1	6.44991	1.02821	0.46084	1.77994	261	254	15.6	58.8	1014.8	0	0	14	56.5	14.6	5.6	10.5026N	76.0120E	05-01-2020	6.23824	118.005
7	30	7.20102	293	2	0	78	39	4	0.09	0.25	107	50	50	0.17864	15	177	100	300	0.16	0.1	6.41358	1.01594	0.56047	1.48959	338	296	17.7	62.79	1014.8	0	0	18.7	69.3	16	5.5	10.5026N	76.0120E	06-01-2020	6.18266	104.962
8	30	7.51534	295	2	0	79	42	4	0.1	0.25	105	51	52	0.067	15	180	100	300	0.16	0.1	4.10236	1.00353	0.43025	0.92348	323	265	18.9	68.91	1015.5	0.2	4.17	40.2	114.6	32.3	4.8	10.5026N	76.0120E	07-01-2020	6.26332	77.6525
9	28	7.46964	308	2	0	76	41	4	0.1	0.25	105	53	52	0.20703	15	172	95	300	0.16	0.1	4.25351	1.02809	0.37036	1.29454	293	254	19.4	68.63	1015.5	0	0	13.6	95	42.5	5.3	10.5026N	76.0120E	08-01-2020	6.06282	94.6084
10	29	7.38162	312	2	0	79	43	4	0.1	0.25	108	51	52	0.57919	17	182	94	300	0.16	0.1	4.44675	1.00294	0.41102	1.09849	313	265	18.3	65.71	1013.7	0	0	14.4	94.9	26.3	5.1	10.5026N	76.0120E	09-01-2020	6.05781	85.6345
11	29	7.33535	293	2	0	80	43	4	0.11	0.25	107	50	53	0.60146	18	179	102	300	0.16	0.1	4.20747	1.00105	0.33374	1.76197	311	260	17.8	63.8	1014.5	0	0	14.9	65.1	14	5.4	10.5026N	76.0120E	10-01-2020	6.06552	116.401
12	29	7.36059	303	2	0	80	40	4	0.11	0.25	106	52	50	0.58709	15	181	96	300	0.16	0.1	4.49398	1.01245	0.53221	0.40795	278	299	17.5	63.33	1014.2	0	0	14	99.1	14.3	5.5	10.5026N	76.0120E	11-01-2020	6.27971	53.9434
13	29	7.27626	306	2	0	78	44	4	0.1	0.25	101	50	53	0.67971	16	178	97	300	0.16	0.1	3.414	1.00475	0.10081	0.90761	334	260	17.3	63.55	1013.4	0	0	14	96.3	12.5	5.7	10.5026N	76.0120E	12-01-2020	6.23449	75.0845
14	27	7.32482	301	2	0	75	43	4	0.1	0.25	107	54	51	0.6144	16	173	101	300	0.16	0.1	3.33575	1.00871	0.27974	0.61184	280	259	17.5	65.59	1013.2	0	0	13.7	84.6	30.8	5.2	10.5026N	76.0120E	13-01-2020	6.02512	61.9335
15	28	7.21468	310	2	0	80	40	4	0.1	0.25	100	53	50	0.79137	17	171	99	300	0.16	0.1	3.42958	1.00575	0.39944	0.50022	326	298	17.4	64.16	1012.9	0	0	13.3	51.8	22.9	5.2	10.5026N	76.0120E	14-01-2020	6.03844	57.482
16	28	7.61377	304	2	0	78	42	4	0.1	0.25	101	51	54	0.51393	17	174	95	300	0.16	0.1	3.17811	1.0109	0.11275	0.3071	324	265	17.3	62.2	1012.7	0	0	11.9	72.7	23.2	5.4	10.5026N	76.0120E	15-01-2020	6.26694	46.9889
17	30	7.68303	293	2	0	78	43	4	0.1	0.25	107	52	52	0.54407	15	171	98	300	0.16	0.1	3.11167	1.01154	0.22019	1.15309	253	282	18.3	65.17	1011.9	0	0	9.4	140.8	26.1	5.2	10.5026N	76.0120E	16-01-2020	6.22315	87.3123
18	28	7.34376	303	2	0	80	43	4	0.11	0.25	103	52	52	0.59134	15	171	101	300	0.16	0.1	3.15035	1.20486	0.15622	0.93151	300	251	18.7	67.23	1011.8	0	0	11.6	100.4	27.4	5.3	10.5026N	76.0120E	17-01-2020	6.02065	76.5922
19	27	7.20223	300	2	0	76	38	4	0.11	0.25	108	52	52	0.71495	15	175	94	300	0.006	0.1	3.43551	1.24263	0.10174	0.96397	268	281	19	65.8	1010.9	0	0	11.6	71.9	33.1	5.6	10.5026N	76.0120E	18-01-2020	6.22533	57.6339
20	30	7.61432	293	2	0	75	44	4	0.1	0.25	104	51	53	0.7133	15	180	94	300	0.006	0.1	3.24455	1.04953	0.11162	0.75816	252	253	20.1	69.3	1010.8	0.5	4.17	9.9	129.3	71.1	5.2	10.5026N	76.0120E	19-01-2020	6.10581	47.7075
21	29	7.64342	291	2	0	78	44	4	0.1	0.25	106	53	50	1.10855	17	171	95	300	0.006	0.1	2.49905	1.14674	0.28513	0.6867	318	252	21	73.02	1012	0	0	14.8	87.6	82	4.8	10.5026N	76.0120E	20-01-2020	6.0922	45.4189
22	28	7.57078	297	2	0	75	39	4	0.09	0.25	104	51	53	1.0125	15	175	100	300	0.006	0.1	2.23057	1.03733	0.29054	0.98596	312	292	21.6	72.69	1012.3	0	0	13	105.3	47.4	5.1	10.5026N	76.0120E	21-01-2020	6.12565	59.5221
23	27	7.26953	303	2	0	79	43	4	0.09	0.25	105	50	53	1.1793	17	182	94	300	0.006	0.1	2.2887	1.14967	0.35259	1.00435	302	270	20	67.26	1013.5	45	8.33	15.5	108.7	34.8	5.3	10.5026N	76.0120E	22-01-2020	6.01659	60.6737
24	27	7.64048	314	2	0	77	43	4	0.09	0.25	100	54	52	1.01589	17	176	101	300	0.006	0.1	2.12016	1.23456	0.20469	0.19692	266	256	20.5	68.02	1013.9	0	0	14.4	125.1	35.6	5	10.5026N	76.0120E	23-01-2020	6.13829	22.0508
25	28	7.3038	287	2	0	76	39	4	0.09	0.25	106	51	54	1.1684	15	171	101	300	0.006	0.1	2.18513	1.25384	0.20742	0.46063	285	269	20.6	68.46	1014.5	0.8	4.17	11.9	50.7	45.9	5.3	10.5026N	76.0120E	24-01-2020	6.08414	34.5053
26	30	7.43363	310	2	0	77	43	4	0.1	0.25	100	54	50	1.22773	18	178	100	300	0.006	0.1	2.03023	1.21057	0.14479	0.48884	350	281	21	71.77	1013.4	0	0	11.2	111.8	54.6	4.8	10.5026N	76.0120E	25-01-2020	6.1436	35.5984
27	28	7.34935	289	2	0	79	45	4	0.1	0.25	108	51	53	1.27058	18	173	100	300	0.006	0.1	2.44925	1.16268	0.33412	0.43998	296	270	20.6	68.65	1013	0.4	4.17	16.2	89	44.5	4.7	10.5026N	76.0120E	26-01-2020	6.10504	34.1095
28	28	7.69411	300	2	0	77	40	4	0.1	0.25	106	51	53	1.06087	16	181	100	300	0.006	0.1	1.32696	1.05056	0.18731	0.21961	335	292	19.9	65.66	1013	0	0	26.5	149.8	41.9	5.1	10.5026N	76.0120E	27-01-2020	6.12745	23.0097
29	29	7.24923	309	2	0	80	45	4	0.11	0.25	102	53	52	1.07526	18	181	102	300	0.006	0.1	1.05173	1.27187	0.2978	0.2717	291	275	18.3	60.58	1013.1	0	0	13.6	83.2	22.4	5.3	10.5026N	76.0120E	28-01-2020	6.04835	26.1296
30	28	7.67139	295	2	0	79	42	4	0.1	0.25	100	53	51	1.27476	17	178	97	300	0.006	0.1	1.43123	1.06368	0.39706	0.4787	301	265	17.7	58.48	1013.4	0	0	15.1	123.5	20.1	5.4	10.5026N	76.0120E	29-01-2020	6.18019	36.1781
31	29	7.53673	296	2	0	76	40	4	0.11	0.25	104	53	51	1.10505	16	181	101	300	0.006	0.1	1.38371	1.28226	0.15748	0.16188	270	297	17.7	60.48	1014.1	0	0	18.5	64.5	24.7	5.6	10.5026N	76.0120E	30-01-2020	6.20915	20.3796
32	29	7.6561	302	2	0	76	41	4	0.1	0.25	106	53	52	1.10002	15	173	99	300	0.006	0.1	1.40383	1.0191	0.35507	0.17974	253	266	16.9	58.42	1013.5	0	0	19.2	55.7	25.7	5.1	10.5026N	76.0120E	31-01-2020	6.24576	21.8464
33	27	7.72131	254	2	0	38	38	4	0.1	0.25	61	33	44	3.03568	14	149	69	300	0.006	0.1	1.05856	1.20067	0.13547	0.2294	306	208	16.5	59.71	1013.8	0	0	15.1	108.7	39.4	5.1	10.5026N	76.0120E	01-02-2020	7.24177	23.2917
34	30	7.32224	197	2	0	33	28	4	0.1	0.25	80	35	33	3.26055	17	123	79	300	0.006	0.1	1.32529	1.12141	0.26286	0.31018	337	209	13.8	51	1014	0	0	18.4	73	26.6	5.3	10.5026N	76.0120E	02-02-2020	7.13333	27.4789
35	27	7.4158	217	2	0	37	37	4	0.1	0.25	62	45	36	3.14509	17	138	61	300	0.006	0.1	1.39168	1.13962	0.23127	0.38911	311	248	15.2	55.03	1013.6	0	0	15.4	59.6	29.8	5.5	10.5026N	76.0120E	03-02-2020	7.03363	31.1278
36	29	7.68856	197	2	0	38	35	4	0.1	0.25	77	49	38	3.21313	14	135	77	300	0.006	0.1	1.49411	1.23042	0.2158	0.48481	276	208	14.6	53.01	1013.2	0	0	20.5	53.2	27.2	5.6	10.5026N	76.0120E	04-02-2020	7.10961	35.6408
37	29	7.34254	240	2	0	36	34	4	0.1	0.25	68	47	46	3.26704	17	143	67	300	0.006	0.1	1.13744	1.09154	0.25639	0.39924	257	204	15.6	56.35	1013.2	0	0	24.1	77.3	19	5.4	10.5026N	76.0120E	05-02-2020	7.20326	31.7097
38	27	7.61543	199	2	0	40	31	4	0.1	0.25	60	50	42	3.23764	16	139	87	300	0.006	0.1	1.07665	1.08894	0.16752	0.48888	320	250	17.4	59.18	1012.2	0	0	14.1	92.8	19.7	5.6	10.5026N	76.0120E	06-02-2020	7.06163	35.5556
39	29	7.52234	189	2	0	40	27	4	0.1	0.25	95	40	42	3.01327	13	117	85	300	0.006	0.1	1.11244	1.04319	0.30006	0.22256	263	239	19.3	62.69	1013	0	0	11.6	97.3	26.9	5.4	10.5026N	76.0120E	07-02-2020	7.05295	23.5894
40	28	7.35682	182	2	0	39	32	4	0.11	0.25	60	35	44	3.22363	15	146	82	300	0.006	0.1	1.40685	1.25364	0.24805	0.12092	311	234	20.2	66.44	1012.2	0	0	11.9	73	34.7	4.9	10.5026N	76.0120E	08-02-2020	7.16234	18.833
41	28	7.68571	186	2	0	43	29	4	0.11	0.25	61	43	39	3.23568	13	126	67	300	0.006	0.1	1.11035	1.25899	0.28445	0.13495	286	219	20.4	65.75	1012.1	0	0	13.4	76	35.9	5	10.5026N	76.0120E	09-02-2020	7.11661	19.5994
42	30	7.74314	215	2	0	45	24	4	0.1	0.25	87	45	50	3.17807	16	111	61	300	0.006	0.1	1.15795	1.14017	0.13367	0.42522	274	210	19	60.78	1011.7	0	0	11.6	163.4	29.9	5.2	10.5026N	76.0120E	10-02-2020	7.23178	32.5341
43	28	7.56035	251	2	0	35	37	4	0.1	0.25	63	42	49	5.20942	15	141	77	300	0.006	0.1	1.4894	1.07227	0.22399	0.38845	316	233	18.4	58.63	1011.3	0	0	13	186.7	21	5.8	10.5026N	76.0120E	11-02-2020	7.18108	31.2118
44	30	7.23489	246	2	0	46	32	4	0.1	0.25	75	33	44	5.22224	17	119	78	300	0.006	0.1	1.34254	1.0501	0.25127	0.3139	255	223	18	58.58	1011.1	0	0	13.6	149.1	15.9	5.8	10.5026N	76.0120E	12-02-2020	7.04046	27.5588
45	30	7.81316	207	2	0	30	26	4	0.1	0.25	65	42	42	5.01903	18	149	88	300	0.006	0.1	1.41306	1.15949	0.16235	0.34282	285	235	18.2	58.86	1011.9	0	0	38.7	169.3	24.8	5.6	10.5026N	76.0120E	13-02-2020	7.00955	28.7095
46	30	7.82844	176	2	0	39	38	4	0.09	0.25	79	40	34	5.18953	18	134	67	300	0.006	0.1	1.23737	1.10067	0.11391	0.16259	263	215	18.4	60.74	1011.6	0	0	18.5	69.3	25.6	5.3	10.5026N	76.0120E	14-02-2020	7.28109	19.818
47	30	7.74197	237	2	0	35	32	4	0.09	0.25	90	47	42	5.09708	14	120	83	300	0.007	0.1	1.17344	1.13883	0.28077	0.48148	295	211	18.5	60.76	1012.7	0	0	17.1	55.4	28	5.8	10.5026N	76.0120E	15-02-2020	7.21409	36.0222
48	30	7.82162	249	2	0	37	33	4	0.09	0.25	87	39	36	5.30402	13	122	86	300	0.007	0.1	1.06936	1.03078	0.38758	0.41277	306	247	19.4	62.84	1013	0	0	16.9	63	37.2	5.7	10.5026N	76.0120E	16-02-2020	7.20075	33.2601
49	28	7.34363	254	2	0	39	34	4	0.1	0.25	65	33	49	5.23392	17	135	79	300	0.007	0.1	1.48191	1.28435	0.21158	0.30096	276	217	19.3	60.18	1012.8	0	0	17.8	58.5	45.3	5.9	10.5026N	76.0120E	17-02-2020	7.2383	27.3177
50	27	7.52272	213	2	0	38	32	8	0.1	0.25	64	42	34	5.28278	17	141	64	300	0.007	0.1	1.03948	1.25699	0.37203	1.06299	265	243	19.2	59.26	1012.7	0	0	22.7	90.3	47.5	5.5	10.5026N	76.0120E	18-02-2020	7.23343	64.3241
51	27	7.41073	255	2	0	35	31	8	0.1	0.25	75	40	36	5.08754	16	147	63	300	0.007	0.1	1.38159	1.26699	0.30106	1.07151	331	208	18.8	57.94	1013.7	0	0	21.5	92.5	33.3	5.9	10.5026N	76.0120E	19-02-2020	7.21574	64.3565
52	28	7.76691	240	2	0	32	29	8	0.1	0.25	84	37	39	7.20807	13	137	90	300	0.007	0.1	1.02101	1.22578	0.25524	1.18586	254	232	20.4	62.14	1013.8	0	0	21.4	133.2	25.7	5.7	10.5026N	76.0120E	20-02-2020	7.17334	69.5143
53	29	7.33009	249	2	0	42	32	8	0.1	0.25	75	41	48	7.23528	14	124	73	300	0.007	0.1	1.35688	1.23289	0.20965	1.08723	250	236	21.2	64.64	1012.5	0	0	16.6	138	27.1	5.8	10.5026N	76.0120E	21-02-2020	7.04323	64.6642
54	28	7.50265	238	2	0	32	24	8	0.1	0.25	62	38	30	7.13347	17	136	91	300	0.007	0.1	1.16659	1.28226	0.36293	1.37708	257	249	21.1	62.42	1012.6	0	0	15.4	139.5	28.3	5.9	10.5026N	76.0120E	22-02-2020	7.13152	79.0053
55	30	7.41922	224	2	0	32	28	8	0.1	0.25	62	43	50	7.25373	14	118	68	300	0.007	0.1	1.19693	1.10403	0.19443	1.01381	323	215	19.7	60.41	1012.2	0	0	14.4	116.3	28.8	5.8	10.5026N	76.0120E	23-02-2020	7.2492	61.0926

WQI-EBP Dataset

APPENDIX B- CODING

Code to calculate WQI using the WQI-PCA dataset

```
importing pandas as pd
import pandas as pd
Read and store content
 of an excel file
read _file = pd.read _excel ("WQI-PCA.xlsx")
Write the dataframe object into csv file
read _file.to _csv ("WQI-PCA.csv",
index = None,
header=True)
read csv file and convert into a dataframe object
df = pd.DataFrame(pd.read _csv("WQI-PCA.csv"))
show the dataframe
df
Temp = df[df.columns[0]]
Temp
i = 1
Temp=[]
for i in range(len(df)):
s1=28
w1=0.004249273
v1=df[df.columns[0]][i]
q1=(v1/s1)*100
tmp = w1*q1
print(tmp)
Temp.append(tmp)
```

```
pH= df[df.columns[1]]
i=1
pH=[]
for i in range(len(df)):
s2=8.5
w2=0.013997604
v2=df[df.columns[1]][i]
q2=(v2/s2)*100
p2 = w2*q2
print(p2)
pH.append(p2)
"----------"for each attribute similar coding
fc= df[df.columns[26]]  # FC
i = 1
fc=[]
for i in range(len(df)):
s27= 60
w27=0.001982994
v27=df[df.columns[26]][i]
q27=(v27/s27)*100
fc27 = w27*q27
print(fc27)
fc.append(fc27)
wqi=[]
total=0
for i in range(len(df)):
total=Temp[i]+pH[i]+cd[i]+tr[i]+pa[i]+ta[i]+ch[i]+cod[i]+tkn[i]+am[i]+ha[i]+cal[i]+mg[i]+sl[i]
+so[i]+tss[i]+tds[i]+fds[i]+ps[i]+b[i]+do[i]+bod[i]+fl[i]+do[i]+ni[i]+tc[i]+fc[i]
wqi.append(total)
```

```
print(wqi)
df = pd.read _csv('WQI-PCA.csv')
df.head()
df["WQI"] = wqi
df. to _csv("WQI-PCA.csv", index=False)
 textbf{Code for Exploratory Data Analysis}
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
get _ipython().run _line _magic('matplotlib', 'inline')
sns.set(color _codes=True)
df = pd.read _csv("weather.csv")
 # To display the top 5 rows
df.head(5)
df.tail(5)
df.dtypes
df.shape
duplicate _rows _df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate _rows _df.shape)
df.count()       # Used to count the number of rows
df.describe()
df = df.drop _duplicates()
df.head(5)
df.count()
print(df.isnull().sum())
df = df.dropna()     # Dropping the missing values.
df.count()
```

```
print(df.isnull().sum())    # After dropping the values
plt.figure(figsize=(10,5))
c= df.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
c
 # Create the default pairplot
sns.pairplot(df)
df.hist()
sns.boxplot(x=df['Temp'])
sns.boxplot(x=df['pH'])
sns.boxplot(x=df['Conductivity '])
sns.boxplot(x=df['Turbidity'])
sns.boxplot(x=df['Total Alkalinity'])
sns.boxplot(x=df['Cholride'])
sns.boxplot(x=df['COD'])
sns.boxplot(x=df['TKN'])
sns.boxplot(x=df['Ammonia'])
sns.boxplot(x=df['Hardness'])
sns.boxplot(x=df['Sodium'])
sns.boxplot(x=df['TDS'])
sns.boxplot(x=df['FDS'])
sns.boxplot(x=df['Pottassium'])
sns.boxplot(x=df['BOD'])
sns.boxplot(x=df['Fluoride'])
sns.boxplot(x=df['Nitrate-N'])
sns.boxplot(x=df['TC'])
sns.boxplot(x=df['Dew'])
sns.boxplot(x=df['Humidity'])
```

```
sns.boxplot(x=df['Sealevelpressure'])
sns.boxplot(x=df['Precipitation"])
sns.boxplot(x=df['Precipcover'])
sns.boxplot(x=df['Windspeed'])
sns.boxplot(x=df['Cloudcover'])
sns.heatmap(c)
sns.heatmap(c,cmap="BrBG")
d=df.corr(method ='pearson')
d
```

Code for Feature Selection

```
from sklearn.datasets import make _regression
from sklearn.model _selection import train _test _split
from sklearn.feature _selection import SelectKBest
from sklearn.feature _selection import f _regression
from matplotlib import pyplot
 # feature selection
def select _features(X _train, y _train, X _test):
 # configure to select all features
fs = SelectKBest(score _func=f _regression, k='all')
 # learn relationship from training data
fs.fit(X _train, y _train)
 # transform train input data
X _train _fs = fs.transform(X _train)
 # transform test input data
X _test _fs = fs.transform(X _test)
return X _train _fs, X _test _fs, fs
 # load the dataset
X, y = make _regression(n _samples=1000, n _features=100, n _informative=10, noise=0.1, random _state=1)
```

```
import pandas as pd
import seaborn as sns
df = pd.read _csv('WQI.csv')
df.head()
 #Divide the features into Independent and Dependent Variable
X = df.drop('WQI' , axis =1)
y = df['WQI']
 # split into train and test sets
X _train, X _test, y _train, y _test = train _test _split(X, y, test _size=0.33, random _state=1)
 # feature selection
X _train _fs, X _test _fs, fs = select _features(X _train, y _train, X _test)
 # what are scores for the features
for i in range(len(fs.scores _)):
print('Feature  %d:  %f'  % (i, fs.scores _[i]))
 # plot the scores
pyplot.bar([i for i in range(len(fs.scores _))], fs.scores _)
pyplot.show()
```

Code for WQI Prediction using GRU using WQI-PCA Dataset

```
import tensorflow as tf
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean _absolute _error, mean _squared _error, r2 _score
from tensorflow.keras.layers import Dense, LSTM, Dropout, GRU
 # load the dataset
df = pd.read _csv("WQI _PCA.csv")
df
 # extract the target variable
```

```
y = df['WQI'].values
 # drop the target variable from the dataset
df = df.drop(['WQI'], axis=1)
 # normalize the input data using MinMaxScaler
scaler = MinMaxScaler()
X = scaler.fit _transform(df)
X
df
 # split the data into training and testing sets
split = int(0.8 * X.shape[0])
X _train, X _test = X[:split, :], X[split:, :]
y _train, y _test = y[:split], y[split:]
 # convert the training and testing sets to constants
X _train = tf.constant(X _train, dtype=tf.float32)
y _train = tf.constant(y _train, dtype=tf.float32)
X _test = tf.constant(X _test, dtype=tf.float32)
y _test = tf.constant(y _test, dtype=tf.float32)
 # add an additional dimension to the input data
X _train = tf.expand _dims(X _train, axis=-1)
X _test = tf.expand _dims(X _test, axis=-1)
 # create the GRU model
model = tf.keras.Sequential()
model.add(tf.keras.layers.GRU(64, input _shape=(X _train.shape[1], 1), return
_sequences=True))
model.add(tf.keras.layers.GRU(32))
model.add(tf.keras.layers.Dense(1, activation='linear'))
model.add(Dropout(0.2))
 # compile the model
model.compile(optimizer='adam', loss='mean _squared _error')
```

```
# fit the model on the training data
history= model.fit(X _train, y _train, epochs=200,validation _split = 0.2, batch _size=64, verbose=2)
# make predictions on the test data
y _pred = model.predict(X _test)
y _pred = y _pred.reshape(-1,)
# calculate the evaluation metrics
mae = mean _absolute _error(y _test, y _pred)
mse = mean _squared _error(y _test, y _pred)
rmse = np.sqrt(mse)
r2 = r2 _score(y _test, y _pred)
# print the evaluation metrics
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2-Score:", r2)
```

Code for WQI Prediction using WQI-SA Dataset and LSTM

```
import pandas
import matplotlib.pyplot as plt
dataset = pandas.read _csv('WQI-SA.csv', usecols=[], engine='python')
# LSTM for Water quality  problem with regression framing
import numpy
import matplotlib.pyplot as plt
from pandas import read _csv
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.metrics import mean _squared _error
 # convert an array of values into a dataset matrix
def create _dataset(dataset, look _back=1):
dataX, dataY = [], []
for i in range(len(dataset)-look _back-1):
a = dataset[i:(i+look _back), 0]
dataX.append(a)
dataY.append(dataset[i + look _back, 0])
return numpy.array(dataX), numpy.array(dataY)
 # fix random seed for reproducibility
numpy.random.seed(7)
 # load the dataset
dataframe = read _csv('weather.csv', usecols=[38], engine='python')
dataset = dataframe.values
dataset = dataset.astype('float32')
 # normalize the dataset
scaler = MinMaxScaler(feature _range=(0, 1))
dataset = scaler.fit _transform(dataset)
 # normalize the dataset
scaler = MinMaxScaler(feature _range=(0, 1))
dataset = scaler.fit _transform(dataset)
 # split into train and test sets
train _size = int(len(dataset) * 0.67)
test _size = len(dataset) - train _size
train, test = dataset[0:train _size,:], dataset[train _size:len(dataset),:]
train, test = dataset[0:train _size,:], dataset[train _size:len(dataset),:]
 # reshape into X=t and Y=t+1
look _back = 1
```

```
trainX, trainY = create _dataset(train, look _back)
testX, testY = create _dataset(test, look _back)
 # reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
 # create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input _shape=(1, look _back)))
model.add(Dense(1))
model.compile(loss='mean _squared _error', optimizer='adam')
model.fit(trainX, trainY, epochs=10, batch _size=1, verbose=2)
 # make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
print(trainPredict.shape)
print(testPredict.shape)
 # invert predictions
trainPredict = scaler.inverse _transform(trainPredict)
trainY = scaler.inverse _transform([trainY])
testPredict = scaler.inverse _transform(testPredict)
testY = scaler.inverse _transform([testY])
y _pred=testPredict
y _test=testX
print( (y _test),(y _pred))
from sklearn import metrics
testScore = metrics.mean _absolute _error(testY[0], testPredict[:,0])
print('Test Score:  %.4f MAE'  % (testScore))
from sklearn import metrics
```

```
 # calculate mean squared error
testScore = metrics.mean _squared _error(testY[0], testPredict[:,0])
print('Test Score:  %.5f MSE'  % (testScore))
 # calculate root mean squared error
testScore = math.sqrt(mean _squared _error(testY[0], testPredict[:,0]))
print('Test Score:  %.4f RMSE'  % (testScore))
from sklearn import metrics
 # calculate r squared
testScore = metrics.r2 _score(testY[0], testPredict[:,0])
print('Test Score:  %.2f R2 score'  % (testScore))
from sklearn import metrics
 # calculate Explained _Variance
testScore = metrics.explained _variance _score(testY[0], testPredict[:,0])
print('Test Score:  %.2f Explained _Variance'  % (testScore))
```

Code for WQI Prediction using TFT

```
EPOCHS = 200
INLEN = 32
HIDDEN = 64
LSTMLAYERS = 2
ATTHEADS = 1
DROPOUT = 0.1
BATCH = 32
N _FC = 12           # default forecast horizon
RAND = 42            # set random state
N _SAMPLES = 100      # number of times a prediction is sampled from a probabilistic model
N _JOBS = 3           # parallel processors to use;  -1 = all processors
 # default quantiles for QuantileRegression
QUANTILES = [0.01, 0.05, 0.1, 0.2, 0.25, 0.5, 0.75, 0.8, 0.9, 0.95, 0.99]
```

```
TRAIN = "20201201"   # train/test split
MSEAS = 12          # max seasonality to check: months
ALPHA = 0.05         # significance level for seasonality test
FIGSIZE = (9, 6)
qL1, qL2, qL3 = 0.01, 0.05, 0.10
  # percentiles of predictions: lower bounds
qU1, qU2, qU3 = 1-qL1, 1-qL2, 1-qL3
 # upper bounds derived from lower bounds
label _q1 = f'{int(qU1 * 100)} / {int(qL1 * 100)} percentile band'
label _q2 = f'{int(qU2 * 100)} / {int(qL2 * 100)} percentile band'
label _q3 = f'{int(qU3 * 100)} / {int(qL3 * 100)} percentile band'
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from darts import TimeSeries, concatenate
from darts.dataprocessing.transformers import Scaler
from darts.models import TFTModel, NaiveSeasonal, NaiveDrift, ExponentialSmoothing
from darts.utils.statistics import check _seasonality, extract _trend _and _seasonality
from darts.metrics import mape
from darts.utils.timeseries _generation import datetime _attribute _timeseries
from darts.utils.likelihood _models import QuantileRegression
from darts.utils.utils import ModelMode, SeasonalityMode, TrendMode
pd.set _option("display.precision",2)
np.set _printoptions(precision=2, suppress=True)
pd.options.display.float _format = '{:,.2f}'.format
df = pd.read _csv('WQI-SA.csv')
series = TimeSeries.from _dataframe(df, 'Month','WQI')
ts = series / TimeSeries.from _series(series.time _index.days _in _month)
```

```
ts = ts.astype(np.float32)
print(type(ts))
df = ts.pd _dataframe()
print(df.shape[0])
print(df.shape[1])
ts = TimeSeries.from _series(df["WQI"])
plt.figure(100, figsize=(18, 5))
ts.plot();
mseas=MSEAS
for m in range(2, MSEAS):
is _seasonal, mseas = check _seasonality(ts, m=m, alpha=ALPHA, max _lag=MSEAS)
if is _seasonal:
break
print("seasonal? " + str(is _seasonal))
if is _seasonal:
print('There is seasonality of order {} months'.format(mseas))
ts _trend, ts _seas = extract _trend _and _seasonality(ts=ts, freq=mseas)
 # train/test
if isinstance(TRAIN, str):
split = pd.Timestamp(TRAIN)
else:
split = TRAIN
ts _train, ts _test = ts.split _after(split)
 #print(ts _train.count())
 #scale the time series on the training settransformer = Scaler()
transformer = Scaler()
ts _ttrain = transformer.fit _transform(ts _train)
print(len(ts _ttrain))
```

```
ts _ttest = transformer.transform(ts _test)
ts _t = transformer.transform(ts)
 # create covariates: year, month, and integer index series
cov = datetime _attribute _timeseries(ts, attribute="year", one _hot=False)
cov = cov.stack(datetime _attribute _timeseries(ts, attribute="month", one _hot=False))
cov = cov.stack(TimeSeries.from _times _and _values(
times=ts.time _index,
values=np.arange(len(ts)),
columns=["linear _increase"]))
cov = cov.astype(np.float32)
 # train/test
train _cov, test _cov = cov.split _after(split)
 # rescale the covariates: fit on the training set
scaler = Scaler()
scaler.fit(train _cov)
tcov = scaler.transform(cov)
 # naive seasonal forecast
modelNs = NaiveSeasonal(K=mseas)
modelNs.fit(ts _train)
ts _predNs = modelNs.predict(len(ts _test))
 # naive drift (trend) forecast
modelNd = NaiveDrift()
modelNd.fit(ts _train)
ts _predNd = modelNd.predict(len(ts _test))
ts _predN = ts _predNd + ts _predNs - ts _train.last _value()
plt.figure(100, figsize=(18, 5))
ts.plot(label="actual")
ts _predN.plot(label="naive forecast")
```

```
plt.title("Naive Forecast (MAPE: {:.2f} %)".format(mape(ts _test, ts _predN)))
plt.legend();
 # search space for best theta value: check 100 alternatives
modelX = ExponentialSmoothing(
seasonal _periods=mseas,
seasonal=ModelMode.MULTIPLICATIVE)
modelX.fit(ts _train)
ts _predX = modelX.predict(  n=len(ts _test),
num _samples=N _SAMPLES)
print(ts _test)
print(ts _predX)
plt.figure(100, figsize=(18, 5))
ts.plot(label="actual")
ts _predX.plot(label="Exponential Smoothing")
plt.title("Exponential Smoothing (MAPE: {:.2f} %)".format(mape(ts _test, ts _predX)))
plt.legend();
 #temporal fusion transformer
model = TFTModel(   input _chunk _length=INLEN,
output _chunk _length=N _FC,
hidden _size=HIDDEN,
lstm _layers=LSTMLAYERS,
num _attention _heads=ATTHEADS,
dropout=DROPOUT,
batch _size=BATCH,
n _epochs=EPOCHS,
likelihood=QuantileRegression(quantiles=QUANTILES),
 # loss _fn=MSELoss(),
random _state=RAND,
```

```
force _reset=True)
 # training
model.fit(  ts _ttrain,
future _covariates=tcov,
verbose=True)
 # testing: generate predictions
ts _tpred = model.predict(   n=len(ts _test),num _samples=N _SAMPLES,n _jobs=N _JOBS)
print("Do the predictions constitute a probabilistic time series?", ts _tpred.is _probabilistic)
 # testing: helper function: plot predictions
def plot _predict(ts _actual, ts _test, ts _pred):
 # plot time series, limited to forecast horizon
plt.figure(figsize=FIGSIZE)
ts _actual.plot(label="actual")
  # plot actual
ts _pred.plot(low _quantile=qL1, high _quantile=qU1, label=label _q1)     # plot U1 quantile band
ts _pred.plot(low _quantile=qL2, high _quantile=qU2, label=label _q2)    # plot U2 quantile band
ts _pred.plot(low _quantile=qL3, high _quantile=qU3, label=label _q3)     # plot U3 quantile band
ts _pred.plot(central _quantile="mean", label="expected")
 # plot "mean" or median=0.5
plt.title("TFT: test set (MAPE: {:.2f} %)".format(mape(ts _test, ts _pred)))
plt.legend();
 # testing: call helper function: plot predictions
ts _pred = transformer.inverse _transform(ts _tpred)
plot _predict(ts, ts _test, ts _pred)
 # testing: call helper function: plot predictions, focus on test set
ts _pred = transformer.inverse _transform(ts _tpred)
ts _actual = ts[ ts _tpred.start _time(): ts _tpred.end _time() ]
 # actual values in forecast horizon
```

```
plot _predict(ts _actual, ts _test, ts _pred)
 # testing: collect the prediction percentiles in a dataframe dfY
dfY = pd.DataFrame()
 # helper method: calculate percentiles of predictions
def predQ(ts _tpred, q):
ts _t = ts _tpred.quantile _timeseries(q)
 # percentile of predictions
ts = transformer.inverse _transform(ts _t)
 # inverse-transform the percentile
s = TimeSeries.pd _series(ts)
header = "Q" + format(int(q*100), "02d")
dfY[header] = s
 # fill Actual column into the new dataframe
dfY["Actual"] = TimeSeries.pd _series(ts _actual)
 # call helper function: percentiles of predictions
quantiles = [0.5, qU1, qU2, qU3, qL3, qL2, qL1]
 _ = [predQ(ts _tpred, q) for q in quantiles]
dfY.iloc[np.r _[0:2, -2:0]]
 # backtest the model: predictions vs actual values
 # start: at 10 % of time series
STARTBACKT = ts.get _timestamp _at _point(point=max(0.2, INLEN))
#pd.Timestamp("19500301")
print("start:",STARTBACKT)
 # forecast horizon: until end of test period
tdiff = (   pd.to _datetime(ts.end _time()).to _period("M") -
pd.to _datetime(STARTBACKT).to _period("M")).n
print("months:",tdiff)
ts _tbacktest = model.historical _forecasts(  series=ts _t,
past _covariates=None,
```

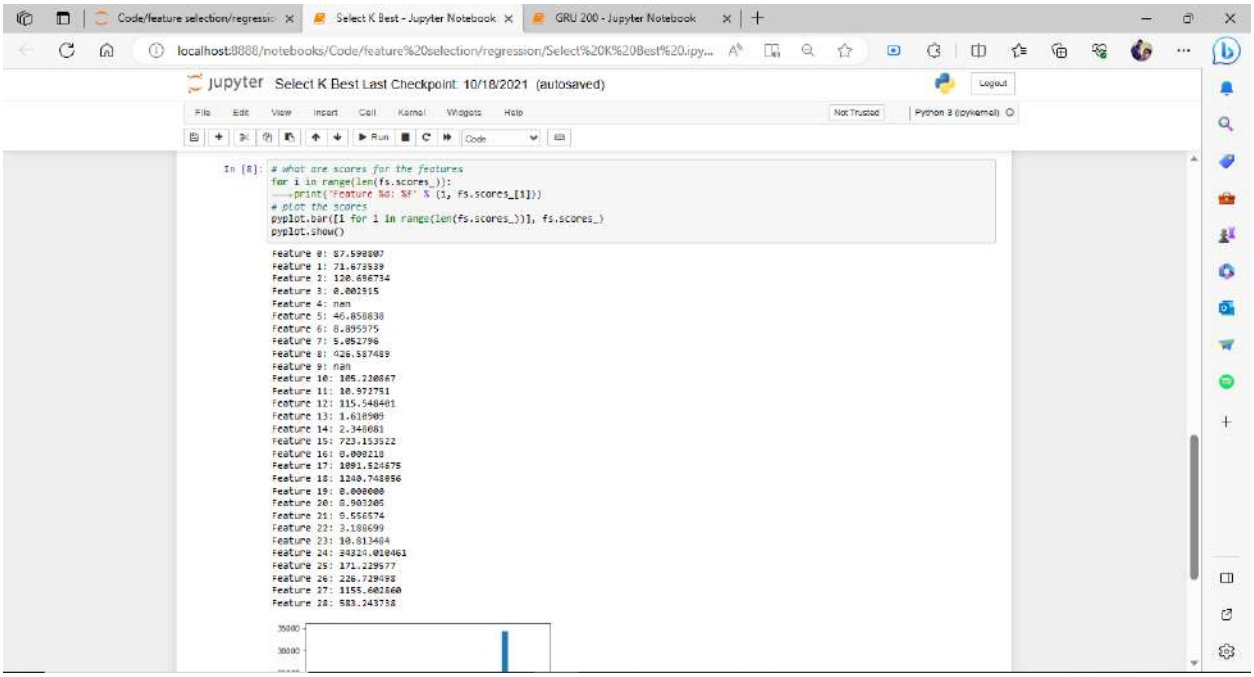
```
future _covariates=tcov,
start=STARTBACKT,
num _samples=N _SAMPLES,
forecast _horizon=tdiff,
stride=12,
last _points _only=False,
retrain=False,
verbose=True)
 # backtesting: helper function: plot backtest predictions
def plot _backtest(ts _backtest, ts _actual, transformer):
plt.figure(figsize=FIGSIZE)
ts _actual.plot(label="actual")
ts _backtest.plot(low _quantile=qL1, high _quantile=qU1, label=label _q1)
ts _backtest.plot(low _quantile=qL3, high _quantile=qU3, label=label _q3)
plt.legend()
MAPE = "MAPE: {:.2f} %".format(mape(ts _actual, ts _backtest))
plt.title(   "TFT: backtest from " +                    format(ts _backtest.start _time(), " %Y. %m") +
" to " + format(ts _backtest.end _time(), " %Y. %m") +                "  (" + MAPE + ")");
 # backtesting: call helper function: plot backtest
ts _backtest = transformer.inverse _transform(ts _tbacktest)
plot _backtest(  ts _backtest=concatenate(ts _backtest),
ts _actual=ts,
transformer=transformer)
 # create future covariates: year, month, integer index
FC _HORIZ = 12
 # months after end of training set
start = ts.start _time()
n _per = len(ts _train) + FC _HORIZ
  # set a maximum horizon to create covariates for
```

```
 # create covariates from beginning of training data to end of forecast horizon
ts _year = datetime _attribute _timeseries(
pd.date _range(start=start, periods=n _per, freq="MS"),   #, closed="right"),
attribute="year",
one _hot=False)
ts _month = datetime _attribute _timeseries(
pd.date _range(start=start, periods=n _per, freq="MS"),
attribute="month",
one _hot=False)
cov = ts _year.stack(ts _month)
 # combine year and month with integer index as third covariate
cov = cov.stack(TimeSeries.from _times _and _values(
times=cov.time _index,
values=np.arange(n _per),
columns=['linear _increase']))
cov = cov.astype(np.float32)
tcov = scaler.transform(cov)
print("start:", cov.start _time(), "; end:",cov.end _time())
 # generate future, out-of-sample predictions
ts _tpred = model.predict(n=FC _HORIZ, future _covariates=tcov, num _samples=N
_SAMPLES)
print("start:", ts _tpred.start _time(), "; end:",ts _tpred.end _time())
ts _pred = transformer.inverse _transform(ts _tpred)
plt.figure(figsize=FIGSIZE)
ts _actual = ts.slice _intersect(other=ts _pred)
ts _actual.plot(label="actual")
ts _pred.plot(low _quantile=qL1, high _quantile=qU1, label=label _q1)
 # plot U1 quantile band
ts _pred.plot(low _quantile=qL3, high _quantile=qU3, label=label _q3)     # plot U3 quantile band
```

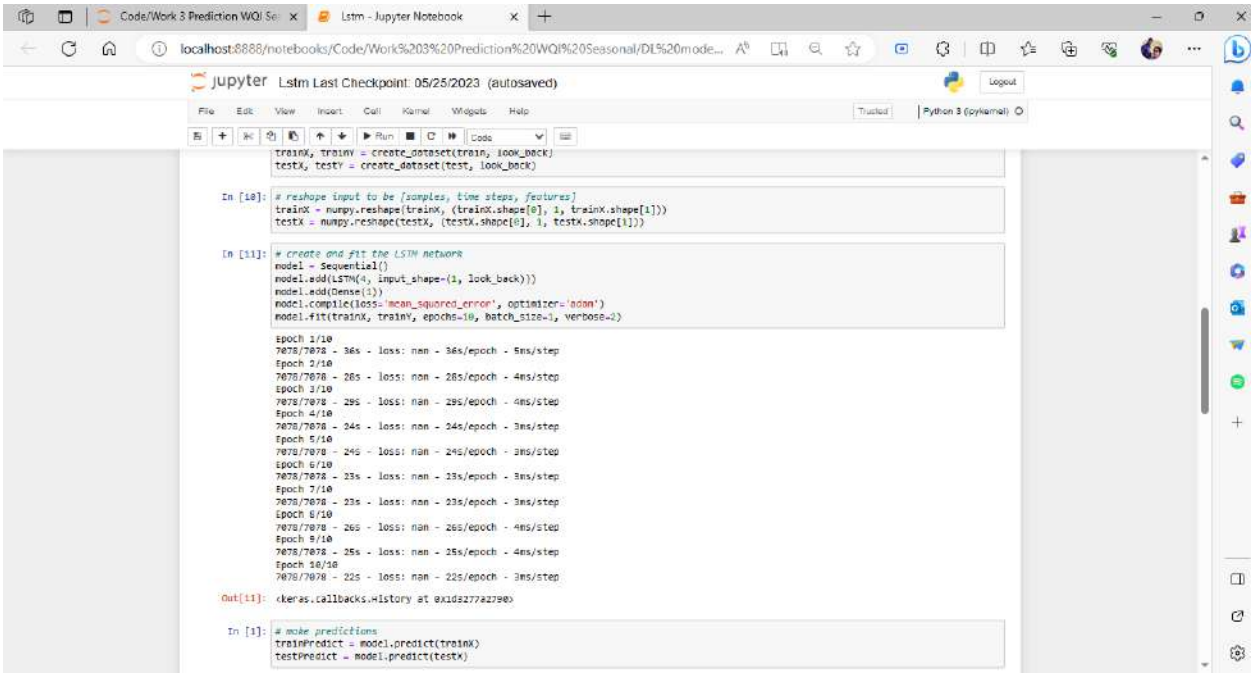
```
ts _pred.plot(central _quantile="mean", label="expected")              # plot "mean" or median=0.5
plt.title(  "TFT: forecast" +           " from " + format(ts _tpred.start _time(), " %Y. %m") +          "
to " + format(ts _tpred.end _time(), " %Y. %m"));
plt.legend();
 # forecast: collect the prediction percentiles in a dataframe
dfY = pd.DataFrame()
 # call helper function predQ: percentiles of predictions
 _ = [predQ(ts _tpred, q) for q in quantiles]
dfY["Actual"] = TimeSeries.pd _series(ts _actual)
dfY.iloc[np.r _[0:2, -2:0]]
ts _tpred.plot(central _quantile="mean", label="expected")              # plot "mean" or median=0.5
plt.legend();
print(ts _actual)
print(ts _pred)
```

APPENDIX C- SCREENSHOTS

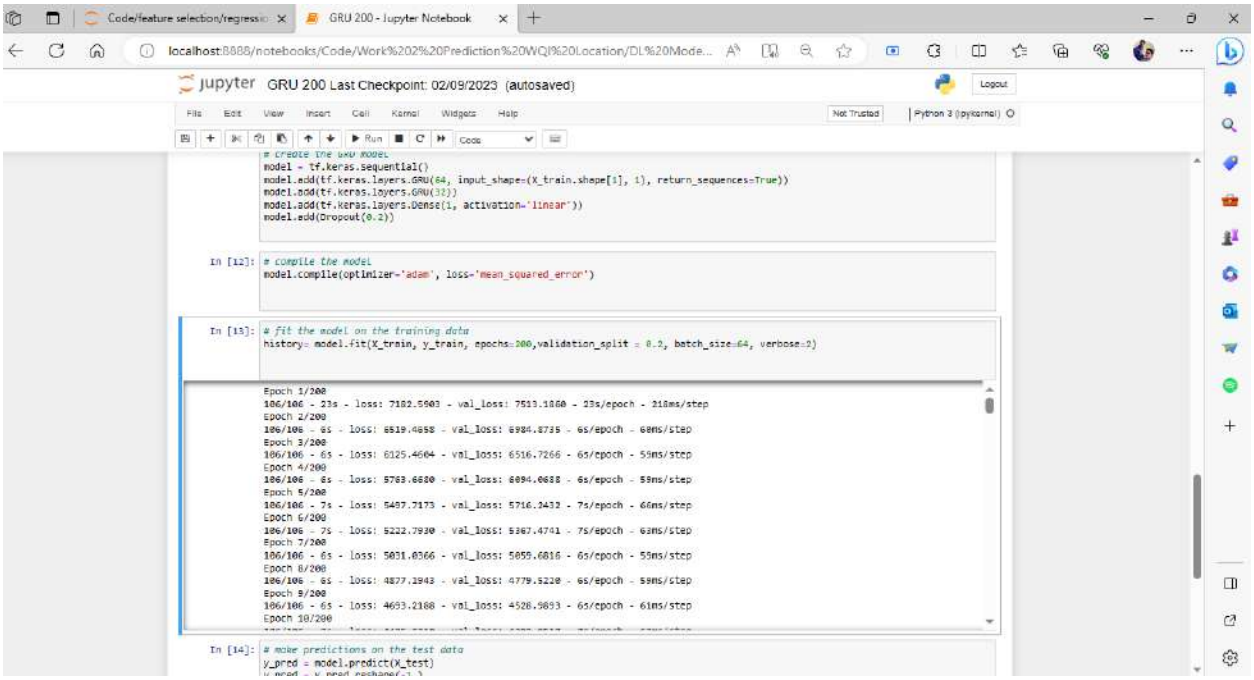
Feature Selection -Select K Best



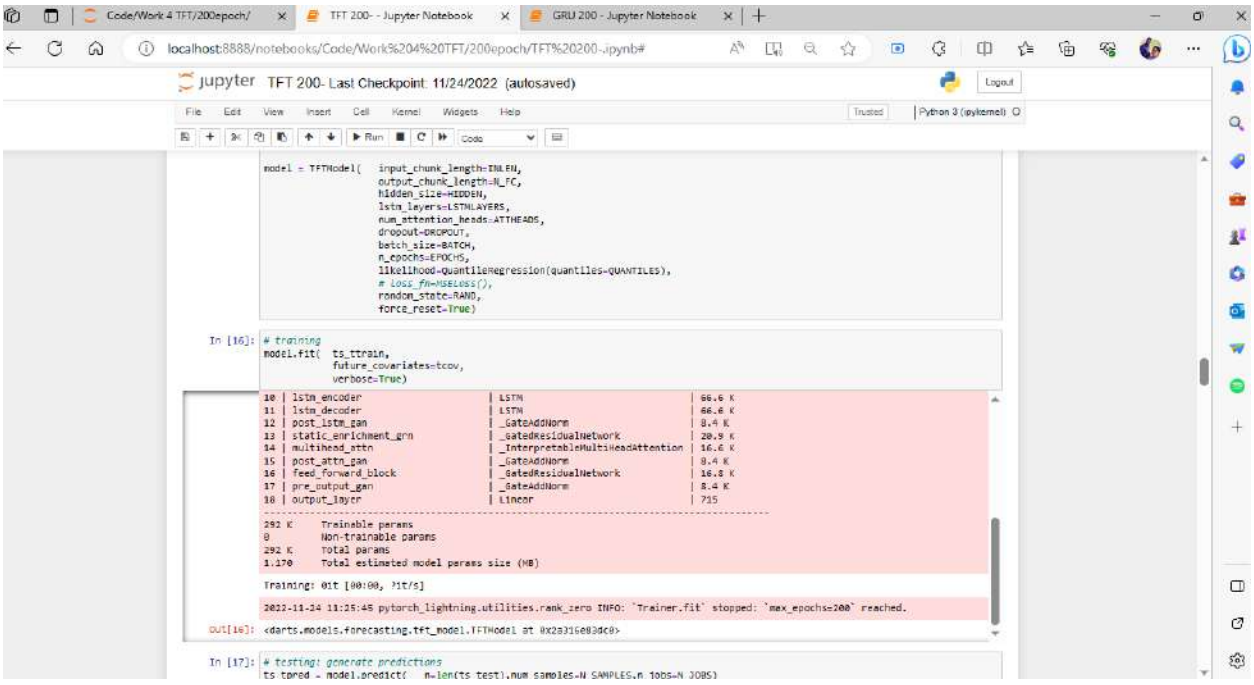
Implementation of LSTM



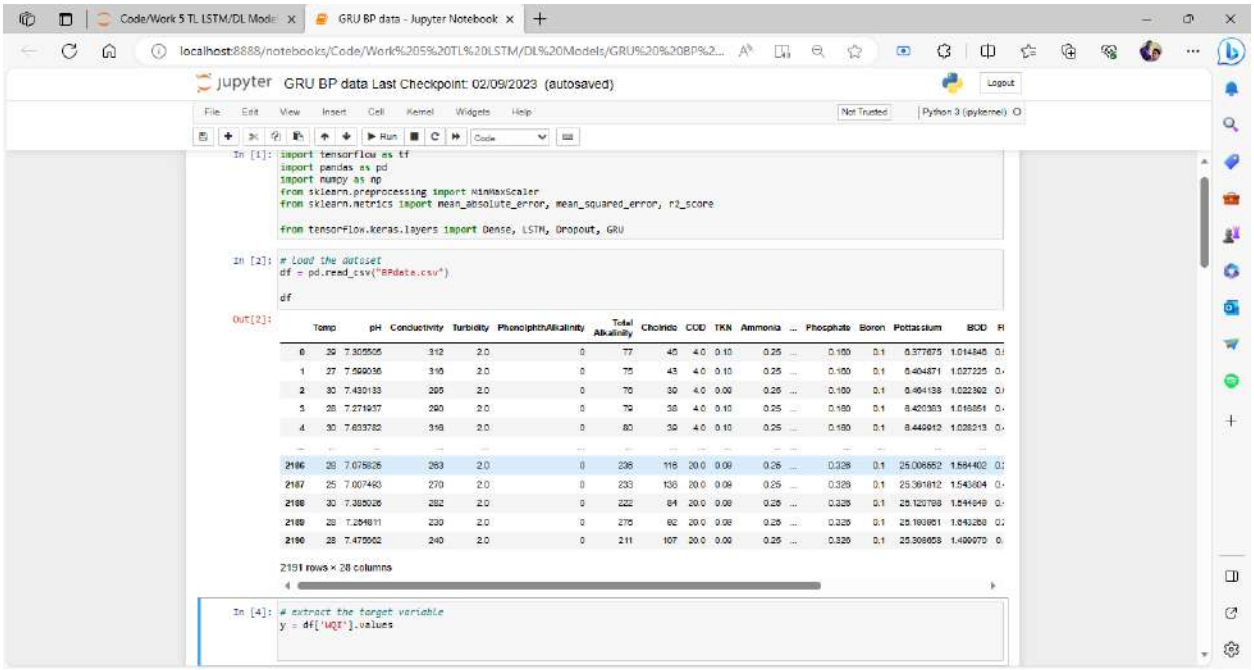
Implementation of GRU



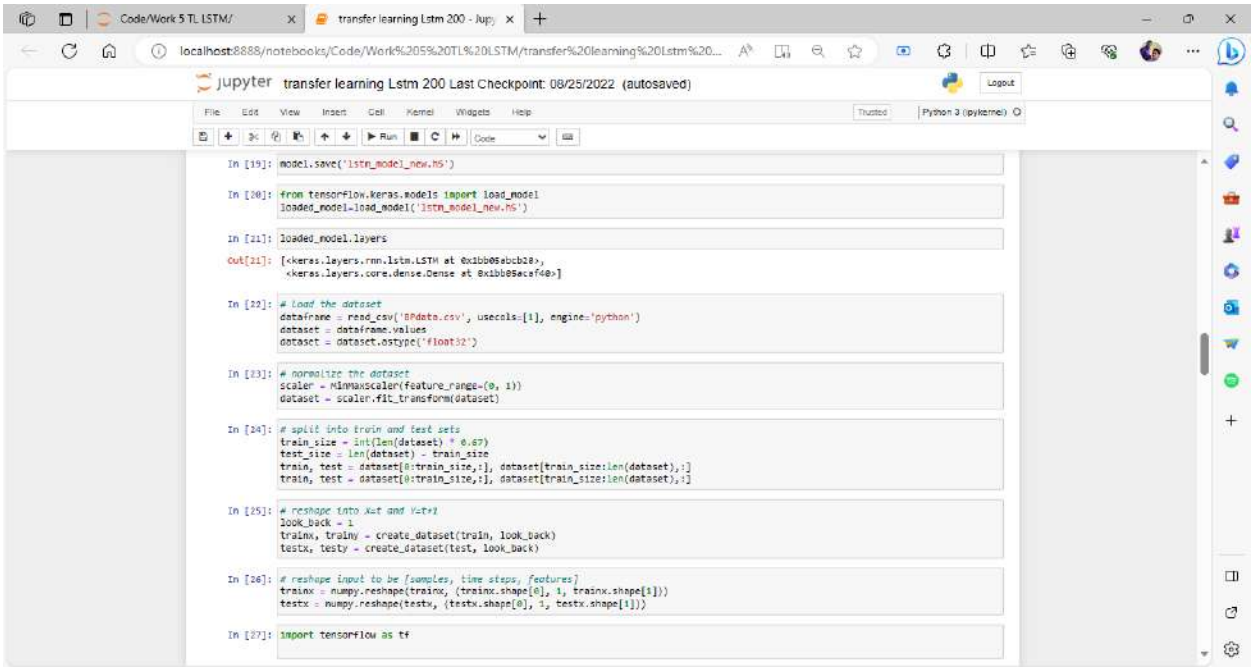
Implementation of Temporal Fusion Transformer



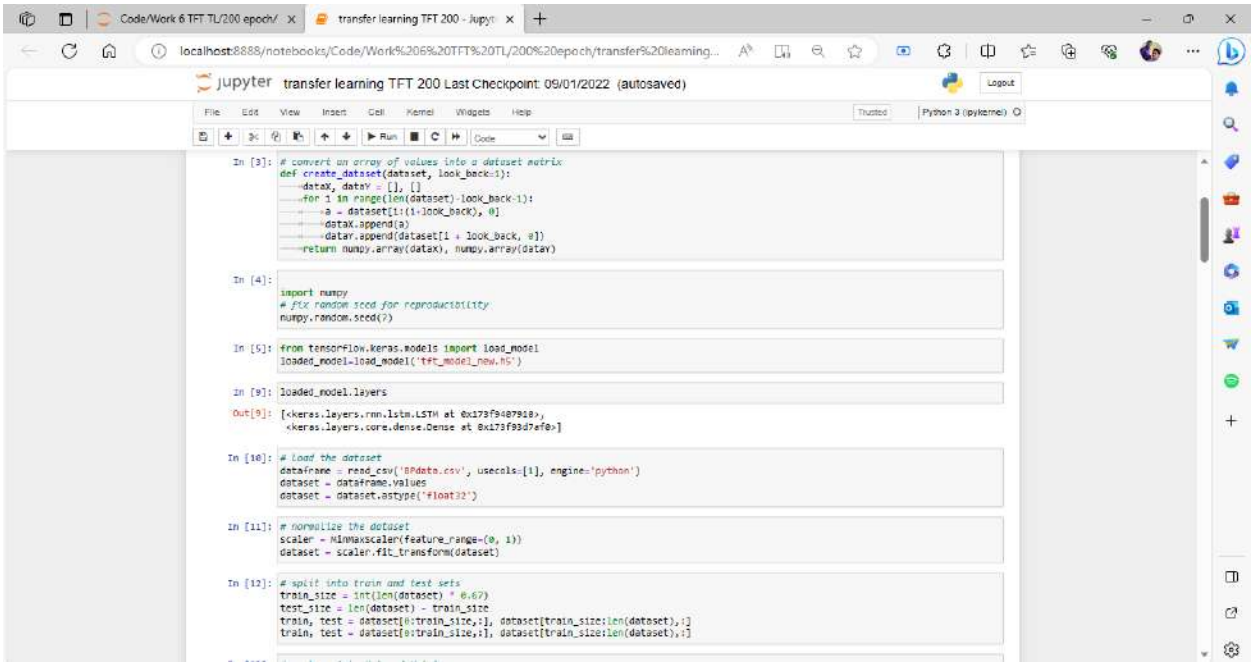
Implementation of GRU with Bharathapuzha Data



Implementation of Transfer Learning with LSTM based Pre-trained Model



Implementation of Transfer Learning with TFT based Pre-trained Model



Implementation of Hetrogenous Transfer Learning with TFT Pre-trained Model