DAFTAR PUSTAKA

Ahmed, B., Kamruzzaman, M. D., Zhu, X., Shahinoor Rahman, M. D., & Choi, K. (2013). Simulating Land Cover Changes and Their Impacts on Land Surface Temperature in Dhaka, Bangladesh. *Remote Sensing*, *5*, 5969–5998. https://doi.org/10.3390/rs5115969

Arsyad, S. (1989). *Konservasi Tanah & Air*. IPB Press.

Azevedo, J. A., Chapman, L., & Muller, C. L. (2016). Quantifying the daytime and night-time urban heat Island in Birmingham, UK: A comparison of satellite derived land surface temperature and high resolution air temperature observations. *Remote Sensing*, *8*(2). https://doi.org/10.3390/rs8020153

Badan Standardisasi Nasional. (2010). Klasifikasi Penutup Lahan. In *SNI 7645:2010*.

Baja, S. (2012). Perencanaan Tata Guna Lahan dalam Pengembangan Wilayah. In *Pendekatan Spasial & Aplikasinya*. Penerbit Andi.

Bappeda. (2015). *Peraturan Daerah Kota Makassar Nomor 4 Tahun 2015 Tentang Rencana Tata Ruang Wilayah Kota Makassar 2015-2034*.

Bappeda Kota Makassar. (2018). *Rencana Program Investasi Jangka Menengah (RPIJM) Bidang Cipta Karya Kota Makassar 2019-2023*.

Bappenas. (2017). *Bunga Rampai Tesis/Disertasi (Tema: Geografi)*. Pusbindiklatren-Bappenas.

BPS Kota Makassar. (2020). *Kota Makassar Dalam Angka Tahun 2020*. BPS Kota Makassar.

Bureau of Environment Tokyo Metropolitan Government. (2018). *Urban Heat Island*. http://www.kankyo.metro.tokyo.jp/en/climate/heat_island.html

Buyantuyev, A., & Wu, J. (2010). Urban Heat Islands and Landscape Heterogeneity: Linking Spatiotemporal Variations in Surface temperatures to Land-Cover and Socioeconomic Patterns. *Landscape Ecology*, *25*, 17–33. https://doi.org/10.1007/s10980-009-9402-4

Crago, R. D., & Qualls, R. J. (2014). Use of Land Surface Temperature to Estimate Surface Energy fluxes : Contributions of Wilfried Brutsaert and Collaborators. *Water Resources Research*, 3396–3408. https://doi.org/10.1002/2013WR015223.Received

Darlina, S. P., Sasmito, B., & Yuwono, B. D. (2018). Analisis Fenomena Urban Heat Island Serta Mitigasinya (Studi Kasus : Kota Semarang). *Jurnal Geodesi Undip*, *7*(3), 77–87.

Darmawan, A., Harianto, S. P., Santoso, T., & Winarno, G. D. (2018). *Buku Ajar Penginderaan Jauh Untuk Kehutanan*.

Deep, S., & Saklani, A. (2014). Urban Sprawl Modeling Using Cellular Automata. *Egyptian Journal of Remote Sensing and Space Science*, *17*, 179–187. https://doi.org/10.1016/j.ejrs.2014.07.001

Deilami, K., Kamruzzaman, M., & Liu, Y. (2018). Urban Heat Island Effect: A Systematic Review of Spatio-temporal Factors, Data, Methods, and Mitigation Measures. *International Journal of Applied Earth Observation and Geoinformation*, *67*, 30–42. https://doi.org/10.1016/j.jag.2017.12.009

Eastman, J. R. (2012). *IDRISI Selva Tutorial*. Idrisi Production.

https://doi.org/10.1016/0301-0104(80)85118-4
Eisavi, V., Homayouni, S., Yazdi, A. M., & Alimohammadi, A. (2015). Land Cover Mapping Based on Random Forest Classification of Multitemporal Spectral and Thermal Images. *Environmental Monitoring and Assessment*, *187*(291), 1–14. https://doi.org/10.1007/s10661-015-4489-3
El-Hattab, M., Amany, S. M., & Lamia, G. E. (2018). Monitoring and assessment of urban heat islands over the Southern region of Cairo Governorate, Egypt. *Egyptian Journal of Remote Sensing and Space Science*, *21*(3), 311–323. https://doi.org/10.1016/j.ejrs.2017.08.008
El Kenawy, A. M., Hereher, M., Robaa, S. M., McCabe, M. F., Moreno, J. I. L., Castro, F. D., Gaber, I. M., Al-Awadhi, T., Al-Buloshi, A., Nasiri, N. Al, Al-Hatrushi, S., Schuwerack, P. M., Angulo, D. P., Abdelaal, M. M., & Serrano, S. M. V. (2020). Nocturnal Surface Urban Heat Island Over Greater Cairo: Spatial Morphology, Temporal Trends and Links to Land-Atmosphere Influences. *Remote Sensing*, *12*(3889), 1–29. https://doi.org/10.3390/rs12233889
EPA. (2014). Reducing Urban Heat Islands : Compendium of Strategies. In *Urban Heat Island Basics*. Climate Protection Partnership Division Environmental Protection Agency's Office.
Ermida, S. L., Soares, P., Mantas, V., Göttsche, F. M., & Trigo, I. F. (2020). Google Earth Engine Open-Source Code for Land Surface Temperature Estimation from the Landsat Series. *Remote Sensing*, *12*(1741), 1–21. https://doi.org/10.3390/RS12091471
ESA. (2018). *Land Surface Temperature - Applications - Sentinel-3 SLSTR - Sentinel Online*. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-3-slstr/overview/geophysical-measurements/land-surface-temperature
Fadillah, D. (2019). Analisis Kerawanan Tanah Longsor Menggunakan Model Integrasi Fuzzy Logic dan Analytical Hierarchy Process (AHP). In *Skripsi*. Universitas Hasanuddin.
FAO. (2016). *Land Cover Classification System*. http://www.fao.org/docrep/003/x0596e/x0596e00.htm
Fardani, I., Mohmed, F. A. J., & Chofyan, I. (2020). Pemanfaatan Prediksi Tutupan Lahan Berbasis Cellular Automata-Markov dalam Evaluasi Rencana Tata Ruang. *MKG*, *21*(2), 157–169.
Farina, A. (2012). Exploring the Relationship Between Land Surface Temperature and Vegetation Abundance for Urban Heat Island Mitigation in Seville, Spain. In *Thesis*. Lund University.
Firozjaei, M. K., Kiavarz, M., Alavipanah, S. K., Lakes, T., & Qureshi, S. (2018). Monitoring and Forecasting Heat Island Intensity Through Multi-Temporal Image Analysis and Cellular Automata-Markov chain Modelling: A case of Babol city, Iran. *Ecological Indicators*, *91*, 155–170. https://doi.org/10.1016/j.ecolind.2018.03.052
Fitriana, A., Subiyanto, S., & Firdaus, H. (2017). Model Cellular Automata Markov Untuk Prediksi Perkembangan Fisik Wilayah Permukiman Kota Surakarta Menggunakan Sistem Informasi Geografis. *Jurnal Geodesi Undip*, *6*(4), 246–253.

Gidey, E., Dikinya, O., Sebego, R., Segosebe, E., & Zenebe, A. (2017). Cellular automata and Markov Chain (CA_Markov) Model-Based Predictions of Future Land Use and Land Cover Scenarios (2015–2033) in Raya, Northern Ethiopia. *Modeling Earth Systems and Environment*. https://doi.org/10.1007/s40808-017-0397-6

Haashemi, S., Weng, Q., Darvishi, A., & Alavipanah, S. K. (2016). Seasonal Variations of the Surface Urban Heat Island in a Semi-Arid City. *Remote Sensing*, *8*(352), 1–17. https://doi.org/10.3390/rs8040352

Hadi, B. S. (2019). *Penginderaan Jauh Digital*. UNY Press. http://staffnew.uny.ac.id/upload/132240452/penelitian/Penginderaan Jauh Pengangtar ke Arah Pembelajaran Berpikir Spasial.pdf

Hakim, P. R., Rahman, A., Suhermanto, & Rachim, E. (2012). Model Koreksi Geometri Sistematik Data Imager Pushbroom Menggunakan Moetode Proyeksi Kolinear. *Jurnal Teknologi Dirgantara*, *10*(2), 121–132.

Hasanlou, M., & Mostofi, N. (2015). Investigating Urban Heat Island Effects and Relation Between Various Land Cover Indices in Tehran City Using Landsat 8 Imagery. *1st International Electronic Conference on Remote Sensing*, *22 June-5 July*, 1–11. https://doi.org/10.3390/ecrs-1-f004

Hedge, N. P., MuraliKrishna, I. V, & ChalapatiRao, K. V. (2009). Study of Cellular Automata Models for Urban Growth. In *Map World Forum*.

Iacono, M., Levinson, D., El-Geneidy, A., & Wasfi, R. (2015). A Markov Chain Model of Land Use Change in the Twin Cities, 1985-2005. *Tema. Journal of Land Use, Mobility and Environment*, *8*(3), 263–276. http://www.tema.unina.it/index.php/tema/article/view/2985

Jatmiko, R. H. (2016). Penggunaan Citra Saluran Inframerah Termal Untuk Studi Perubahan Liputan Lahan dan Suhu Sebagai Indikator Perubahn Iklim Perkotaan di Yogyakarta. In *Disertasi*. Universitas Gadjah Mada.

Jensen, J. R. (2015). *Introductory Digital Image Processing A Rempte Sensing Perspective 4th Edition* (4th ed.). Pearson Education.

Keeratikasikorn, C., & Bonafoni, S. (2018). Urban heat island analysis over the land use zoning plan of Bangkok by means of Landsat 8 imagery. *Remote Sensing*, *10*(3). https://doi.org/10.3390/rs10030440

Kurniati, R. (2019). Pengaruh Ketersediaan Ruang Terbuka Hijau Terhadap Urban Heat Island di Kota Makassar, Sulawesi Selatan. In *Thesis*. Universitas Gadjah Mada.

Kurniati, R., & Rahmi, D. H. (2020). Ketersediaan Ruang Terbuka Hijau dan Urban Heat Island di Kota Makassar. *Jurnal Litbang Sukowati*, *3*(2), 1–14.

Kurniawan, M. I. (2016). Analisis Geospasial Terhadap Konversi Lahan Dengan Menggunakan Metode Cellular Automata. In *Skripsi*. Universitas Hasanuddin.

Kurniawan, W. D. W. (2019). Probabilitas Perubahan Tutupan Lahan Berdasarkan Keberadaan Lokasi Wisata di Wilayah Pesisir Sarbagita. *SPACE*, *1*(1), 33–39.

Kushardono, D. (2017). *Klasifikasi Digital Pada Penginderaan jauh*. Penerbit IPB Press. https://doi.org/10.1007/1-4020-0613-6_5006

Kusumadewi, S., & Guswaludin, I. (2005). Fuzzy Multi-Criteria Decision Making. *Media Informatika*, *3*(1), 25–38.

Li, H., Zhou, Y., Li, X., Meng, L., Wang, X., Wu, S., & Sodoudi, S. (2018). A new

method to quantify surface urban heat island intensity. *Science of the Total Environment*. https://doi.org/10.1016/j.scitotenv.2017.11.360

Lillesand, T. M., Kiefer, R. W., & Chipman, J. W. (2015). Remote Sensing and Image Interpretation. In *Photogrammetric Engineering & Remote Sensing* (7th ed., Vol. 81, Issue 8). John Wiley & Sons. https://doi.org/10.14358/pers.81.8.615

Liu, Y. (2009). *Modelling Urban Development with Geographical Information Systems and Cellular Automata*. Taylor & Francis Group.

Lu, L., Weng, Q., Xiao, D., Guo, H., Li, Q., & Hui, W. (2020). Spatiotemporal Variation of Surface Urban Heat Islands in Relation to Land Cover Composition and Configuration: A Multi-Scale Case Study of Xi'an, China. *Remote Sensing*, *12*(2713), 1–19. https://doi.org/10.3390/RS12172713

Lukiawan, R., Purwanto, E. H., & Ayundyahrini, M. (2019). Analisis Pentingnya Standar Koreksi Geometrik Citra Satelit Resolusi Menengah Dan Kebutuhan Manfaat Bagi Pengguna. *Jurnal Standardisasi*, *21*(1), 45–54. https://doi.org/10.31153/js.v21i1.735

Macarof, P., & Statescu, F. (2017). Comparasion of NDBI and NDVI as Indicators of Surface Urban Heat Island Effect in Landsat 8 Imagery: A Case Study of Iasi. *Present Environment and Sustainable Development*, *11*(2), 141–150. https://doi.org/10.1515/pesd-2017-0032

Malik, M. S., Shukla, J. P., & Mishra, S. (2019). Relationship of LST, NDBI and NDVI Using Landsat-8 Data in Kandaihimmat Watershed, Hoshangabad, India. *Indian Journal of Geo-Marine Sciences*, *48*(1), 25–31.

Marsitha, F., Pattipeilohy, W. J., & Virgianto, R. H. (2019). Kenyamanan Termal Klimatologis Kota-Kota Besar Di Pulau Sulawesi Berdasarkan Temperature Humidity Index (Thi). *Jurnal Saintika UNPAM*, *1*(2), 202–211. https://doi.org/10.32493/jsmu.v1i2.2384

Maru, R., Baharuddin, I. I., Umar, R., Rasyid, R., Uca, Sanusi, W., & Bayudin. (2015). Analysis of The Heat Island Phenomenon in Makassar, South Sulawesi, Indonesia. *American Journal of Applied Sciences*. https://doi.org/10.3844/ajassp.2015

Mirnayani, Rapang, S. K., Aini, A. N., & Bayanuddin, A. A. (2021). Pemanfaatan Data Citra Sentinel-3 Sea and Land Surface Temperature Radiometer (SLSTR) Pagi dan Malam Hari Untuk Analisis Intensitas Fenomena Pulau Bahang Permukaan (Studi Kasus : Kota Bandung). *Jurnal Penginderaan Jauh*, *18*(1), 15–25.

Moser, G., Serpico, S. B., & Benediktsson, J. A. (2013). Land-Cover Mapping by Markov Modeling of Spatial-Contextual Information in Very-High-Resolution Remote Sensing Images. *Proceedings of the IEEE*, *101*(3), 631–651. https://doi.org/10.1109/JPROC.2012.2211551

Mushore, T. D., Odindi, J., Dube, T., & Mutanga, O. (2017). Prediction of Future Urban Surface Temperatures Using Medium Resolution Satellite Data in Harare Metropolitan City, Zimbabwe. *Building and Environment*. https://doi.org/10.1016/j.buildenv.2017.06.033

NASA. (2020a). *Landsat Homepage | Landsat Science*. https://landsat.gsfc.nasa.gov/

NASA. (2020b). *What Is an Urban Heat Island? | NASA Climate Kids*. https://climatekids.nasa.gov/heat-islands/

NOAA. (2018). *What is the Difference Between Land Cover and Land Use?* National Oceanic and Atmospheric Administration. http://oceanservice.noaa.gov/facts/lclu.html

Noi Phan, T., Kuch, V., & Lehnert, L. W. (2020). Land Cover Classification Using Google Earth Engine and Random Forest Classifier-The Role of Image Composition. *Remote Sensing*, *12*(2411), 1–22. https://doi.org/10.3390/RS12152411

Paharuddin. (2012). Simulasi Geospasial Berbasis Cellular Automata Perubahan Penggunaan Lahan Untuk Prediksi Sedimentasi. In *Disertasi*. Universitas Hasanuddin.

Parhusip, J. (2019). Penerapan Metode Analytical Hierarchy Process (AHP) Pada Desain Sistem Pendukung Keputusan Pemilihan Calon Penerima Bantuan Pangan Non Tunai (BPNT) Di Kota Palangka Raya. *Jurnal Teknologi Informasi*, *13*(2), 18–29. https://doi.org/10.47111/jti.v13i2.251

Peng, S., Piao, S., Ciais, P., Friedlingstein, P., Ottle, C., Bréon, F. M., Nan, H., Zhou, L., & Myneni, R. B. (2012). Surface urban heat island across 419 global big cities. *Environmental Science and Technology*. https://doi.org/10.1021/es2030438

Peruge, T. V. D. (2013). Model Perubahan Penggunaan Lahan Menggunakan Cellular Automata-Markov Chain di Kawasan Mamminasata. In *Skripsi*. Universitas Hasanuddin.

Pichierri, M., Bonafoni, S., & Biondi, R. (2012). Satellite air temperature estimation for monitoring the canopy layer heat island of Milan. *Remote Sensing of Environment*. https://doi.org/10.1016/j.rse.2012.08.025

Purwadhi, S. H., & Sanjoto, T. B. (2008). *Pengantar Interpretasi Citra Penginderaan Jauh*. Pusat Data Penginderaan Jauh LAPAN dan Geografi Universitas Negeri Semarang.

Rahmadewi, D. P. (2019). Kajian Perubahan Penutup Lahan Dengan Pemodelan Cellular Automata dan Pengaruhnya Terhadap Suhu Permukaan Lahan di Kabupaten Semarang. In *Skripsi*. Universitas Negeri Semarang.

Rahmadewi, D. P., & Hanafi, F. (2020). Kajian Perubahan Penutup Lahan Dengan Pemodelan Cellular Automata dan Pengaruhnya Terhadap Suhu Permukaan Lahan di Kabupaten Semarang. *Geo Image*, *9*(2), 154–166.

Rahmah, A. N., Subiyanto, S., & Amarrohman, F. J. (2020). Pemodelan Perubahan Penggunaan Lahan Dengan Artificial Neural Network (ANN) di Kota Semarang. *Jurnal Geodesi Undip*, *9*(1), 197–206.

Rahman, M. T., Aldosary, A. S., & Mortoja, M. G. (2017). Modeling Future Land Cover Changes and Their Effects on the Land Surface Temperatures in the Saudi Arabian Eastern Coastal City of Dammam. *Land*, *6*(36), 1–16. https://doi.org/10.3390/land6020036

Ravanelli, R., Nascetti, A., Cirigliano, R. V., Di Rico, C., Leuzzi, G., Monti, P., & Crespi, M. (2018). Monitoring the impact of land cover change on surface urban heat island through Google Earth Engine: Proposal of a global methodology, first applications and problems. *Remote Sensing*, *10*(9), 1–21.

https://doi.org/10.3390/rs10091488
Renard, F., Alonso, L., Fitts, Y., Hadjiosif, A., & Comby, J. (2019). Evaluation of the effect of urban redevelopment on surface urban heat islands. *Remote Sensing*, *11*(3), 1–31. https://doi.org/10.3390/rs11030299
Rocha, J., Ferreira, J. C., Simões, J., & Tenedório, J. A. (2007). Modelling Coastal and Land Use Evolution Patterns Through Neural Network and Cellular Automata Integration. *Journal of Coastal Research*, 827–831.
Saaty, R. W. (1987). The Analytic Hierarchy Process-What It Is and How It Is Used. *Mathematical Modelling*, *9*(3–5), 161–176. https://doi.org/10.1016/0270-0255(87)90473-8
Sambodo, K. A., Rahayu, M. I., Indriasari, N., & Natsir, M. (2014). Klasifikasi Hutan-Non Hutan Data Alos Palsar Menggunakan Metode Random Forest. *Prosiding Seminar Nasional Penginderaan Jauh 2014*, 120–127.
Sejati, A. W., Buchori, I., & Rudiarto, I. (2019). The Spatio-Temporal Trends of Urban Growth and Surface Urban Heat Islands Over Two Decades in the Semarang Metropolitan Region. *Sustainable Cities and Society*. https://doi.org/10.1016/j.scs.2019.101432
Sharifi, E., & Lehmann, S. (2015). Correlation Analysis of Surface Temperature of Rooftops, Streetscapes and Urban Heat Island Effect: Case Study of Central Sydney. *Journal of Urban and Environmental Engineering*, *9*(1), 3–11. https://doi.org/10.4090/juee.2015.v9n1.003011
Shastri, H., Barik, B., Ghosh, S., Venkataraman, C., & Sadavarte, P. (2017). Flip flop of Day-night and Summer-Winter Surface Urban Heat Island Intensity in India. *Scientific Reports*, *7*(40178), 1–8. https://doi.org/10.1038/srep40178
Shorabeh, S. N., Hamzeh, S., Shahraki, S. Z., Firozjaei, M. K., & Arsanjani, J. J. (2020). Modelling the Intensity of Surface Urban Heat Island and Predicting the Emerging Patterns: Landsat Multi-Temporal Images and Tehran As Case Study. *International Journal of Remote Sensing*, *41*(19), 7384–7410. https://doi.org/10.1080/01431161.2020.1759841
Shumilo, L., Kussul, N., Shelestov, A., Korsunska, Y., & Yailymov, B. (2019). Sentinel-3 Urban Heat Island Monitoring and analysis for Kyiv Based on Vector Data. *2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 131–135. https://doi.org/10.1109/DESSERT.2019.8770042
Sobirin, & Fatimah, R. N. (2015). Urban Heat Island Kota Surabaya. *Geoedukasi*, *4*(2), 46–69.
Sobrino, J. A., Oltra-Carrió, R., Sòria, G., Bianchi, R., & Paganini, M. (2012). Impact of spatial resolution and satellite overpass time on evaluation of the surface urban heat island effects. *Remote Sensing of Environment*, *117*, 50–56. https://doi.org/10.1016/j.rse.2011.04.042
Sobrino, José Antonio, & Irakulis, I. (2020). A methodology for comparing the surface urban heat Island in selected urban agglomerations around the world from Sentinel-3 SLSTR data. *Remote Sensing*, *12*(12). https://doi.org/10.3390/RS12122052
Song, Y., & Wu, C. (2016). Examining the impact of Urban Biophysical Composition and Neighboring Environment on Surface Urban Heat Island

Effect. *Advances in Space Research*, *57*, 96–109. https://doi.org/10.1016/j.asr.2015.10.036

Stathopoulou, M., & Cartalis, C. (2009). Downscaling AVHRR Land Surface Temperatures for Improved Surface Urban Heat Island Intensity Estimation. *Remote Sensing of Environment*, 2592–2605. https://doi.org/10.1016/j.rse.2009.07.017

Sugandi, D. (2010). *Bahan Pembelajaran Penginderaan Jauh*. http://file.upi.edu/Direktori/FPIPS/JUR._PEND._GEOGRAFI/195805261986031-DEDE_SUGANDI/Sil_PJ.pdf

Suriana, D., Barkey, R. A., & Gou, Z. (2020). Analysis of Land Use/Land Cover Change And Their Effects On Spatiotemporal Patterns Of Urban Heat Islands (UHI) In The City Of Makassar , Indonesia. *International Journal of Engineering and Science Applications*, *7*(2), 113–123.

Susilo, B. (2011). Pemodelan Spasial Probabilistik Integrasi Markov Chain Dan Cellular Automata Untuk Kajian Perubahan Penggunaan Lahan Skala Regional Di Provinsi Daerah Istimewa Yogyakarta. *Jurnal Geografi Gea*, *11*(2), 163–178. https://doi.org/10.17509/gea.v11i2.1638

Syamsuddin, I., & Hwang, J. (2009). The Application of AHP Model to Guide Decision Makers: A Case Study of E-Banking Security. *ICCIT 2009 - 4th International Conference on Computer Sciences and Convergence Information Technology*. https://doi.org/10.1109/ICCIT.2009.251

Tan, J., Yu, D., Li, Q., Tan, X., & Zhou, W. (2020). Spatial Relationship Between Land-Use/Land-Cover Change and Land Surface Temperature in The Dongting Lake Area, China. *Scientific Reports*, *10*(9245), 1–9. https://doi.org/10.1038/s41598-020-66168-6

Tariq, A., & Shu, H. (2020). CA-Markov Chain Analysis of Seasonal Land Surface Temperature and Land Use Land Cover Change Using Optical Multi-Temporal Satellite Data of Faisalabad, Pakistan. *Remote Sensing*, *12*(3402), 1–22. https://doi.org/10.3390/rs12203402

To'sambo, Y. (2018). Analisis Geospasial Prediksi Ketersediaan Lahan Permukiman Kota Makassar Menggunakan Metode Cellular Automata. In *Skripsi*. Universitas Hasanuddin.

Ullah, S., Ahmad, K., Sajjad, R. U., Abbasi, A. M., Nazeer, A., & Tahir, A. A. (2019). Analysis and Simulation of Land Cover Changes and Their Impacts on Land Surface Temperature in A Lower Himalayan Region. *Journal of Environmental Management*, *245*, 348–357. https://doi.org/10.1016/j.jenvman.2019.05.063

Ullah, S., Tahir, A. A., Akbar, T. A., Hassan, Q. K., Dewan, A., Khan, A. J., & Khan, M. (2019). Remote Sensing-Based Quantification of The Relationships Between Land Use Land Cover Changes and Surface Temperature Over The Lower Himalayan region. *Sustainability*, *11*(5492), 1–16. https://doi.org/10.3390/su11195492

United Nations. (2015). *Transforming Our World : The 2030 Agenda for Sustainable Development*. http://www.un.org/

United Nations. (2018). *2018 Revision of World Urbanization Prospects | Multimedia Library - United Nations Department of Economic and Social*

Affairs. United Nations. https://www.un.org/development/desa/publications/2018-revision-of-world-urbanization-prospects.html

USGS. (2016). *Landsat 8 Data Users Handbook*. https://landsat.usgs.gov/documents/Landsat8DataUsersHandbook.pdf

Wang, K., Jiang, S., Wang, J., Zhou, C., Wang, X., & Lee, X. (2017). Comparing the diurnal and seasonal variabilities of atmospheric and surface urban heat islands based on the Beijing urban meteorological network. *Journal of Geophysical Research*, *122*(4), 2131–2154. https://doi.org/10.1002/2016JD025304

Wang, S. Q., Zheng, X. Q., & Zang, X. B. (2012). Accuracy Assessments of Land Use Change Simulation Based on Markov-Cellular Automata Model. *Procedia Environmental Sciences*, *13*, 1238–1245. https://doi.org/10.1016/j.proenv.2012.01.117

Wardana, I. K. (2015). *Analysis of Urban Surface Temperature For Green Spaces Planning In Bandung City, Indonesia*. University of Twente.

Weng, Q. (2009). Thermal Infrared Remote Sensing for Urban Climate and Environmental Studies: Methods, applications, and Trends. *ISPRS Journal of Photogrammetry and Remote Sensing*, *64*, 335–344. https://doi.org/10.1016/j.isprsjprs.2009.03.007

Weng, Q., Firozjaei, M. K., Sedighi, A., Kiavarz, M., & Alavipanah, S. K. (2018). Statistical Analysis of Surface Urban Heat Island Intensity Variations: A Case Study of Babol city, Iran. *GIScience and Remote Sensing*, 1–29. https://doi.org/10.1080/15481603.2018.1548080

WHO. (2012). WHO | Urbanization and health. In *Who*. http://www.who.int/globalchange/ecosystems/urbanization/en/

Widiastusi, J. (2018). Klasifikasi Pembiayaan Warung Mikro Menggunakan Metode Random Forest dengan Teknik Sampling Kelas Imbalanced. In *Skripsi*. Universitas Islam Indonesia.

Wiweka. (2014). Pola Suhu Permukaan dan Udara Menggunakan Citra Satelit Landsat Multitemporal. *Jurnal Ecolab*, *8*(1), 11–22. https://doi.org/10.20886/jklh.2014.8.1.11-22

Wulder, M. A., White, J. C., Loveland, T. R., Woodcock, C. E., Belward, A. S., Cohen, W. B., Fosnight, E. A., Shaw, J., Masek, J. G., & Roy, D. P. (2015). The Global Landsat Archive: Status, Consolidation, and Direction. *Remote Sensing of Environment*, 1–13. https://doi.org/10.1016/j.rse.2015.11.032

Yao, N., Huang, C., Yang, J., Bosch, C. C. K. van den, Ma, L., & Jia, Z. (2020). Combined Effects of Impervious Surface Change and Large-scale Afforestation on the Surface Urban Heat Island Intensity of Beijing, China Based on Remote Sensing Analysis. *Remote Sensing*, *12*(3906), 1–22. https://doi.org/10.3390/rs12233906

Yu, Z., Guo, X., Zeng, Y., Koga, M., & Vejre, H. (2018). Variations in Land Surface Temperature and Cooling Efficiency of Green Space in Rapid Urbanization: The Case of Fuzhou city, China. *Urban Forestry and Urban Greening*, *29*, 113–121. https://doi.org/10.1016/j.ufug.2017.11.008

Yuan, Y., Xi, C., Jing, Q., & Felix, N. (2017). Seasonal Variations of the Urban

Thermal Environment Effect in a Tropical Coastal City. *Advances in Meteorology*, *2017*. https://doi.org/10.1155/2017/8917310

Yudarwati, R., Sitorus, S. R. ., & Munibah, K. (2016). Arahan Pengendalian Perubahan Penggunaan Lahan Menggunakan Markov - Cellular Automata Di Kabupaten Cianjur. *Tataloka*, *18*(4), 211–221. https://doi.org/10.14710/tataloka.18.4.211-221

Zhou, B., Rybski, D., & Kropp, J. P. (2017). The Role of City Size and Urban Form in The Surface Urban Heat Island. *Scientific Reports*, *7*(4791), 1–9. https://doi.org/10.1038/s41598-017-04242-2

Zhou, D., Xiao, J., Bonafoni, S., Berger, C., Deilami, K., Zhou, Y., Frolking, S., Yao, R., Qiao, Z., & Sobrino, J. A. (2019). Satellite remote sensing of surface urban heat islands: Progress, challenges, and perspectives. *Remote Sensing*, *11*(1), 1–36. https://doi.org/10.3390/rs11010048

LAMPIRAN

LAMPIRAN 1 : Pengolahan Data Tutupan Lahan

1. Script Klasifikasi Tutupan Lahan Tahun 2010

LULC Classification L7 10 | Get Link | Save | Run | Reset | Apps

```
Imports (6 entries)
  var geometry2: Polygon, 4 vertices
  var vegetasi_rapat: FeatureCollection (300 elements)
  var vegetasi_jarang: FeatureCollection (300 elements)
  var lahan_terbangun: FeatureCollection (300 elements)
  var perairan: FeatureCollection (300 elements)
  var table: Table users/Mirnayani/shpmakassar
```

```
//load Landsat 7 Tier 1 TOA image collection
var MIN_SCALE = 1/3;
var MAX_SCALE = 3;
var MIN_NEIGHBORS = 144;

/* Apply the USGS L7 Phase-2 Gap filling protocol, using a single kernel size. */
var GapFill = function(src, fill, kernelSize, upscale) {
  var kernel = ee.Kernel.square(kernelSize * 30, "meters", false)

  // Find the pixels common to both scenes.
  var common = src.mask().and(fill.mask())
  var fc = fill.updateMask(common)
  var sc = src.updateMask(common)
  Map.addLayer(common.select(0).mask(common.select(0)), {palette:['000000']}, 'common mask (both exist)', false)

  // Find the primary scaling factors with a regression.
  // Interleave the bands for the regression.  This assumes the bands have the same names.
  var regress = fc.addBands(sc)

  regress = regress.select(regress.bandNames().sort())

  var ratio = 5

  if(upscale) {
    var fit = regress
      .reduceResolution(ee.Reducer.median(), false, 500)
      .reproject(regress.select(0).projection().scale(ratio, ratio))
      .reduceNeighborhood(ee.Reducer.linearFit().forEach(src.bandNames()), kernel, null, false)
      .unmask()
      .reproject(regress.select(0).projection().scale(ratio, ratio))
  } else {
Missing semicolon.egress
      .reduceNeighborhood(ee.Reducer.linearFit().forEach(src.bandNames()), kernel, null, false)
  }

  var offset = fit.select(".*_offset")
  var scale = fit.select(".*_scale")

  Map.addLayer(scale.select('B1_scale'), {min:-2, max:2}, 'scale B1', false)

  // Find the secondary scaling factors using just means and stddev
  var reducer = ee.Reducer.mean().combine(ee.Reducer.stdDev(), null, true)

  if(upscale) {
    var src_stats = src
      .reduceResolution(ee.Reducer.median(), false, 500)
      .reproject(regress.select(0).projection().scale(ratio, ratio))
      .reduceNeighborhood(reducer, kernel, null, false)
      .reproject(regress.select(0).projection().scale(ratio, ratio))

    var fill_stats = fill
      .reduceResolution(ee.Reducer.median(), false, 500)
      .reproject(regress.select(0).projection().scale(ratio, ratio))
      .reduceNeighborhood(reducer, kernel, null, false)
      .reproject(regress.select(0).projection().scale(ratio, ratio))
  } else {
    var src_stats = src
      .reduceNeighborhood(reducer, kernel, null, false)

    var fill_stats = fill
      .reduceNeighborhood(reducer, kernel, null, false)
  }

  var scale2 = src_stats.select(".*stdDev").divide(fill_stats.select(".*stdDev"))
  var offset2 = src_stats.select(".*mean").subtract(fill_stats.select(".*mean").multiply(scale2))

  var invalid = scale.lt(MIN_SCALE).or(scale.gt(MAX_SCALE))
  Map.addLayer(invalid.select(0).mask(invalid.select(0)), {palette:['550000']}, 'invalid1', false)
  scale = scale.where(invalid, scale2)
  offset = offset.where(invalid, offset2)

  // When all else fails, just use the difference of means as an offset.
  var invalid2 = scale.lt(MIN_SCALE).or(scale.gt(MAX_SCALE))
  Map.addLayer(invalid2.select(0).mask(invalid2.select(0)), {palette:['552020']}, 'invalid2', false)
  scale = scale.where(invalid2, 1)
  offset = offset.where(invalid2, src_stats.select(".*mean").subtract(fill_stats.select(".*mean")))
```

```
  // Apply the scaling and mask off pixels that didn't have enough neighbors.
  var count = common.reduceNeighborhood(ee.Reducer.count(), kernel, null, true, "boxcar")
  var scaled = fill.multiply(scale).add(offset)
      .updateMask(count.gte(MIN_NEIGHBORS))

  return src.unmask(scaled, true)
}

var source = ee.Image("LANDSAT/LE07/C01/T1_TOA/LE07_114064_20100902")
var fill = ee.Image("LANDSAT/LE07/C01/T1_TOA/LE07_114064_20100310")

/*
Map.addLayer(source, {bands: ['B3', 'B2', 'B1'], min: 0, max: 0.3}, 'Makassar_2020src');
Map.addLayer(fill, {bands: ['B3', 'B2', 'B1'], min: 0, max: 0.3}, 'Makassar_2020fill');
*/

/*
Map.addLayer(fill, {min:0, max:0.3, bands:["B3", "B2", "B1"]}, "fill", true)
Map.addLayer(source, {min:0, max:0.3, bands:["B3", "B2", "B1"]}, "destination", true)
*/

var landsat7_2010 = GapFill(source, fill, 10, false);
Map.addLayer(landsat7_2010, {min:0, max:0.3, bands:["B3", "B2", "B1"]}, "filled", true)

var landsat7_2010 = GapFill(source, fill, 10, true);
Map.addLayer(landsat7_2010, {min:0, max:0.3, bands:["B3", "B2", "B1"]}, "filled (upscaled)", true)

/*
Map.addLayer(landsat7_2010, {bands: ['B7', 'B5', 'B3'], min: 0, max: 0.3}, 'Urban', true);
Map.addLayer(landsat7_2010, {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.3}, 'Vegetation', true);
*/


//Merge sample points together into one FeatureCollection
var landcover_2010 = vegetasi_rapat.merge(vegetasi_jarang).merge(lahan_terbangun).merge(perairan);

//Select Bands from mosaic Image for training
var bands = ['B1', 'B2', 'B3', "B4", 'B5', 'B6_VCID_1', 'B6_VCID_2', 'B7'];

//The name of the property on the points storing the class lebel
var classProperty = 'landcover';

//Sample the input imagery to get a FeatureCollection of training data
var training = landsat7_2010.select(bands).sampleRegions({
  collection: landcover_2010,
  properties: [classProperty],
  scale: 30
});

//Train the classifier
var classifier = ee.Classifier.smileRandomForest(10).train({
  features: training,
  classProperty: classProperty,
});

//Classify the input imagery
var classified_2010 = landsat7_2010.classify(classifier);

//Define color palette
var palette = [
  '04831F', // vegetasi_rapat (0) // green {htmlcolorcodes.com}
  '23F506', // vegetasi_jarang (1)
  'F30707', //lahan terbangun (2) // red
  '0724ed' //perairan (3) // blue
];

//Display the classified result
Map.addLayer(classified_2010, {
  min:0, max: 3, palette: palette}, 'LULC_2010');

//Optinally, do some accuracy assessment. First, add a column of random uniforms to the training dataset.
var withRandom = training.randomColumn('random');

//We want to reserve some of the data for testing, to avoid overfitting the model.
var split = 0.7; //Roughly 70% training, 30% testing.
var trainingPartition = withRandom.filter(ee.Filter.lt('random', split));
var testingPartition = withRandom.filter(ee.Filter.gte('random', split));

// Get a confusion matrix representing resubstitution accuracy.
var trainAccuracy = classifier.confusionMatrix();
print('Resubstitution error matrix: ', trainAccuracy);
print('Training overall accuracy: ', trainAccuracy.accuracy());

//classify the test FeatureCollection.
var test = testingPartition.classify(classifier);

// Get a confusion matrix representing expected accuracy.
var testAccuracy = test.errorMatrix(classProperty, 'classification');
print('Validation error matrix: ', testAccuracy);
print('Validation overall accuracy: ', testAccuracy.accuracy());
```

```
/*
Export.image.toDrive({
  image: classified_2010,
  description: 'NewData_Tutupanlahan_2010',
  scale: 30,
  region: table,
  fileFormat: 'GeoTIFF',
});
*/
```

2. Script Klasifikasi Tutupan Lahan tahun 2015

LULC Classification L8 15 | Get Link | Save | Run | Reset | Apps

```
Imports (6 entries)
  var geometry2: Polygon, 4 vertices
  var vegetasi_rapat: FeatureCollection (300 elements)
  var vegetasi_jarang: FeatureCollection (300 elements)
  var lahan_terbangun: FeatureCollection (300 elements)
  var perairan: FeatureCollection (300 elements)
  var table: Table users/Mirnayani/shpmakassar
//load Landsat 8 Tier 1 TOA image collection
var landsat_2020 = ee.ImageCollection("LANDSAT/LC08/C01/T1_TOA")
                .filterDate("2015-07-22", "2015-07-31")
                .filter(ee.Filter.eq('WRS_PATH', 114))
                .filter(ee.Filter.eq('WRS_ROW', 64))
                .filter(ee.Filter.lt('CLOUD_COVER',10))
                .sort('CLOUD_COVER')
                .first()
                .clip(geometry2);

print(landsat_2020);


Map.addLayer(landsat_2020, {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.3}, 'Citra_2020');

//Visualisasi citra
Map.addLayer(landsat_2020, {bands: ['B7', 'B6', 'B4'], min: 0, max: 0.3}, 'Urban');
Map.addLayer(landsat_2020, {bands: ['B5', 'B4', 'B3'], min: 0, max: 0.3}, 'Vegetation');
Map.addLayer(landsat_2020, {bands: ['B6', 'B5', 'B2'], min: 0, max: 0.3}, 'Agriculture');
Map.addLayer(landsat_2020, {bands: ['B5', 'B6', 'B4'], min: 0, max: 0.3}, 'Water');


//Merge sample points together into one FeatureCollection
var landcover_2020 = vegetasi_rapat.merge(vegetasi_jarang).merge(lahan_terbangun).merge(perairan);

//Select Bands from mosaic Image for training
var bands = ['B2', 'B3', "B4", 'B5', 'B7'];

//The name of the property on the points storing the class lebel
var classProperty = 'landcover';

//Sample the input imagery to get a FeatureCollection of training data
var training_2020 = landsat_2020.select(bands).sampleRegions({
  collection: landcover_2020,
  properties: [classProperty],
  scale: 30
});

//Train the classifier
var classifier = ee.Classifier.smileRandomForest(10).train({
  features: training_2020,
  classProperty: classProperty,
});

//Classify the input imagery
var classified_2020 = landsat_2020.classify(classifier);

//Define color palette
var palette = [
  '04831F', // vegetasi_rapat (0) // green {htmlcolorcodes.com}
  '23F506', // vegetasi_jarang (1)
  'F30707', //lahan terbangun (2) // red
  '0724ed' //perairan (3) // blue
];

//Display the classified result
Map.addLayer(classified_2020, {
  min:0, max: 3, palette: palette}, 'LULC_2020');


//Optinally, do some accuracy assessment. First, add a column of random uniforms to the training dataset.
var withRandom = training_2020.randomColumn('random');

//We want to reserve some of the data for testing, to avoid overfitting the model.
var split = 0.7; //Roughly 70% training, 30% testing.
var trainingPartition = withRandom.filter(ee.Filter.lt('random', split));
var testingPartition = withRandom.filter(ee.Filter.gte('random', split));
```

```
// Get a confusion matrix representing resubstitution accuracy.
var trainAccuracy_2020 = classifier.confusionMatrix();
print('Resubstitution error matrix_2020: ', trainAccuracy_2020);
print('Training overall accuracy_2020: ', trainAccuracy_2020.accuracy());

//classify the test FeatureCollection.
var test_1 = testingPartition.classify(classifier);

// Get a confusion matrix representing expected accuracy.
var testAccuracy_2020 = test_1.errorMatrix(classProperty, 'classification');
print('Validation error matrix_2020: ', testAccuracy_2020);
print('Validation overall accuracy_2020: ', testAccuracy_2020.accuracy());

/*
//Determine the Land Cover Change between 2001 and 2019
var LULC_Change = classified_2020.subtract(classified_2015);
Map.addLayer(LULC_Change.clip(table), {
  min:0, max:2, palette: ['green', 'red', 'blue']}, 'LULC Change Between 2015 and 2020_Makassar');
*/


/*
Export.image.toDrive({
  image: classified_2020,
  description: 'NewData_Tutupanlahan_2015',
  scale: 30,
  region: table,
  fileFormat: 'GeoTIFF',
});
*/
```

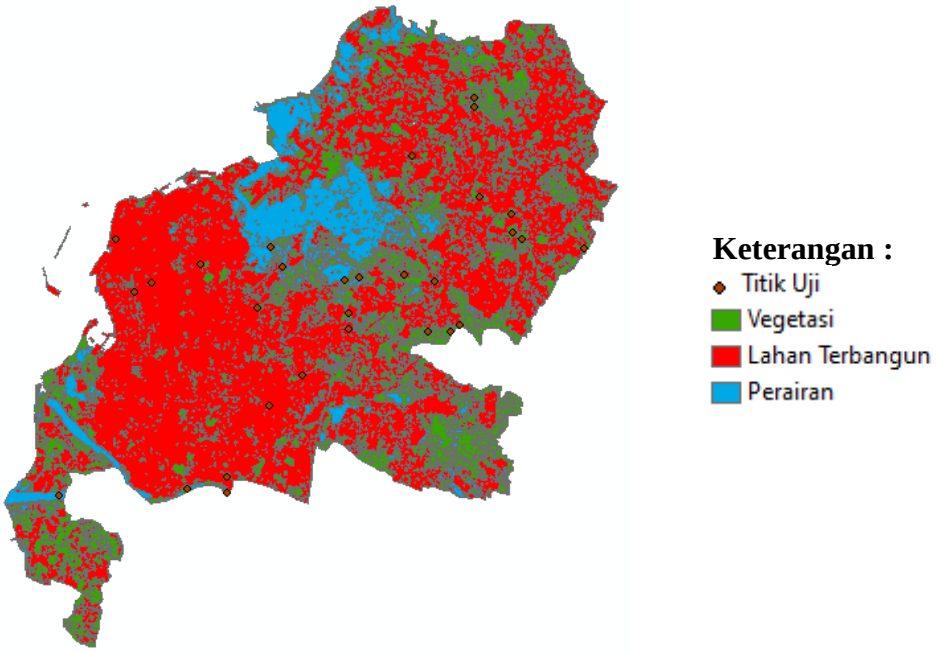
3. Script Klasifikasi Tutupan Lahan tahun 2020

LULC Classification L8 20 | Get Link | Save | Run | Reset | Apps

```
Imports (6 entries)
  var geometry2: Polygon, 4 vertices
  var vegetasi_rapat: FeatureCollection (300 elements)
  var vegetasi_jarang: FeatureCollection (300 elements)
  var lahan_terbangun: FeatureCollection (300 elements)
  var perairan: FeatureCollection (300 elements)
  var table: Table users/Mirnayani/shpmakassar
//load Landsat 8 Tier 1 TOA image collection
var landsat_2020 = ee.ImageCollection("LANDSAT/LC08/C01/T1_TOA")
                .filterDate("2020-08-01", "2020-12-31")
                .filter(ee.Filter.eq('WRS_PATH', 114))
                .filter(ee.Filter.eq('WRS_ROW', 64))
                .filter(ee.Filter.lt('CLOUD_COVER',10))
                .sort('CLOUD_COVER')
                .first()
                .clip(geometry2);

print(landsat_2020);


Map.addLayer(landsat_2020, {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.3}, 'Citra_2020');

//Visualisasi citra
Map.addLayer(landsat_2020, {bands: ['B7', 'B6', 'B4'], min: 0, max: 0.3}, 'Urban');
Map.addLayer(landsat_2020, {bands: ['B5', 'B4', 'B3'], min: 0, max: 0.3}, 'Vegetation');
Map.addLayer(landsat_2020, {bands: ['B6', 'B5', 'B2'], min: 0, max: 0.3}, 'Agriculture');
Map.addLayer(landsat_2020, {bands: ['B5', 'B6', 'B4'], min: 0, max: 0.3}, 'Water');


//Merge sample points together into one FeatureCollection
var landcover_2020 = vegetasi_rapat.merge(vegetasi_jarang).merge(lahan_terbangun).merge(perairan);

//Select Bands from mosaic Image for training
var bands = ['B2', 'B3', "B4", 'B5', 'B7'];

//The name of the property on the points storing the class lebel
var classProperty = 'landcover';

//Sample the input imagery to get a FeatureCollection of training data
var training_2020 = landsat_2020.select(bands).sampleRegions({
  collection: landcover_2020,
  properties: [classProperty],
  scale: 30
});

//Train the classifier
var classifier = ee.Classifier.smileRandomForest(10).train({
  features: training_2020,
  classProperty: classProperty,
});

//Classify the input imagery
var classified_2020 = landsat_2020.classify(classifier);
```

```
//Define color palette
var palette = [
  '04831F', // vegetasi_rapat (0) // green {htmlcolorcodes.com}
  '23F506', // vegetasi_jarang (1)
  'F30707', //lahan terbangun (2) // red
  '0724ed' //perairan (3) // blue
];

//Display the classified result
Map.addLayer(classified_2020, {
  min:0, max: 3, palette: palette}, 'LULC_2020');


//Optinally, do some accuracy assessment. First, add a column of random uniforms to the training dataset.
var withRandom = training_2020.randomColumn('random');

//We want to reserve some of the data for testing, to avoid overfitting the model.
var split = 0.7; //Roughly 70% training, 30% testing.
var trainingPartition = withRandom.filter(ee.Filter.lt('random', split));
var testingPartition = withRandom.filter(ee.Filter.gte('random', split));

// Get a confusion matrix representing resubstitution accuracy.
var trainAccuracy_2020 = classifier.confusionMatrix();
print('Resubstitution error matrix_2020: ', trainAccuracy_2020);
print('Training overall accuracy_2020: ', trainAccuracy_2020.accuracy());

//classify the test FeatureCollection.
var test_1 = testingPartition.classify(classifier);

// Get a confusion matrix representing expected accuracy.
var testAccuracy_2020 = test_1.errorMatrix(classProperty, 'classification');
print('Validation error matrix_2020: ', testAccuracy_2020);
print('Validation overall accuracy_2020: ', testAccuracy_2020.accuracy());

/*
//Determine the Land Cover Change between 2001 and 2019
var LULC_Change = classified_2020.subtract(classified_2015);
Map.addLayer(LULC_Change.clip(table), {
  min:0, max:2, palette: ['green', 'red', 'blue']}, 'LULC Change Between 2015 and 2020_Makassar');
*/


/*
Export.image.toDrive({
  image: classified_2020,
  description: 'NewData_Tutupanlahan_2020',
  scale: 30,
  region: table,
  fileFormat: 'GeoTIFF',
});
*/
```

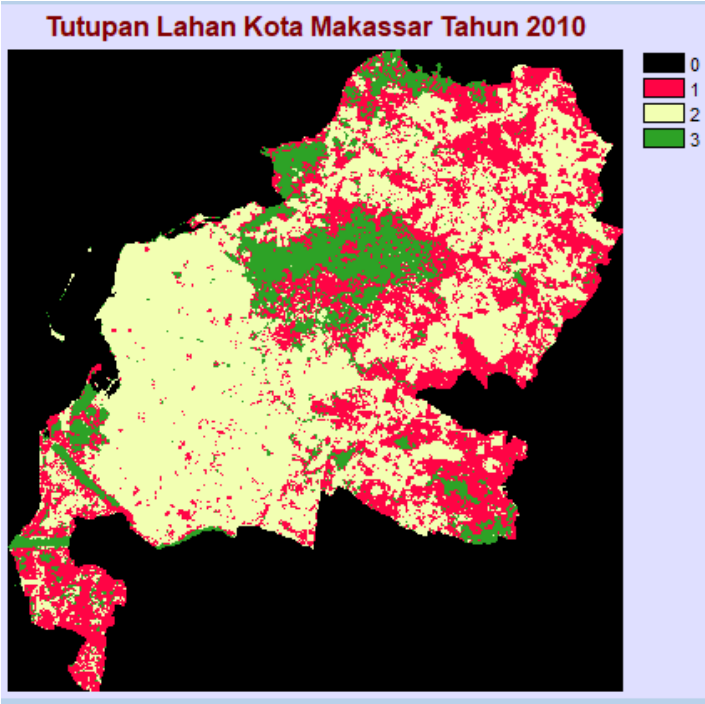
4. Titik Sampel Uji Akurasi Tutupan Lahan tahun 2010, 2015, 2020

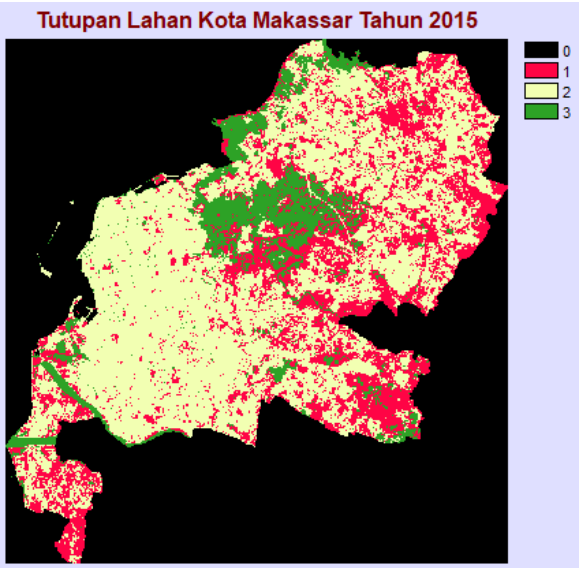


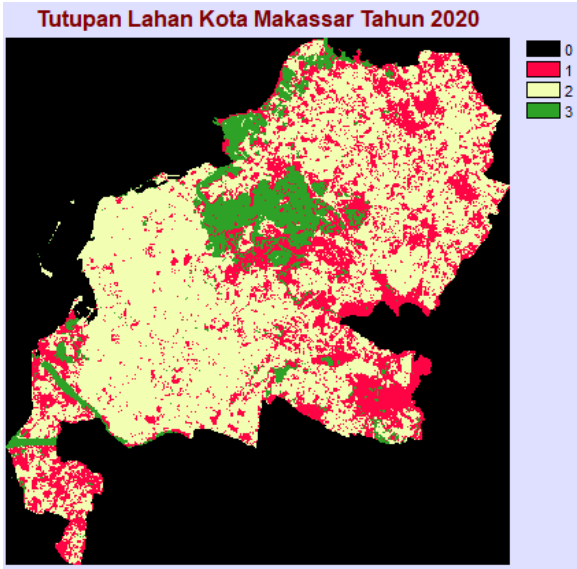
Titik Uji

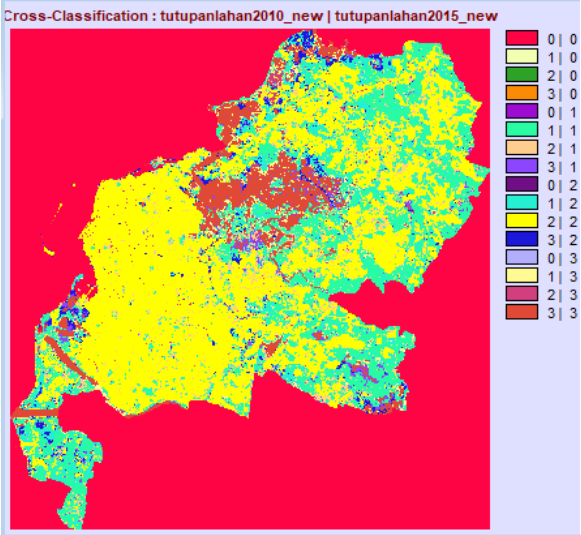
FID	Shape *	NAMOBJ	Validasi	User10	User15	User20
20	Point ZM	Vegetasi	1	1	1	1
21	Point ZM	Vegetasi	1	1	1	1
22	Point ZM	Sawah	1	1	1	1
23	Point ZM	Sawah	1	1	1	1
24	Point ZM	Sawah	1	3	1	1
25	Point ZM	Sawah	1	1	1	1
26	Point ZM	Sawah	1	1	2	1
27	Point ZM	Sawah	1	1	1	1
28	Point ZM	Sawah	1	1	1	1
29	Point ZM	Vegetasi	1	2	1	1
0	Point ZM	Monumen Maha Putra Emi Saelan	2	2	2	2
1	Point ZM	Monumen Mandala	2	2	2	2
2	Point ZM	Monumen 40000	2	2	2	2
10	Point ZM	PLTU Tello	2	2	2	2
14	Point ZM	RS Wahidin	2	2	2	2
15	Point ZM	Pelabuhan Soekarno Hatta	2	2	2	2
16	Point ZM	Terminal Daya	2	2	2	2
17	Point ZM	Terminal Malengkeri	2	2	2	2
18	Point ZM	Universitas Hasanuddin	2	2	2	1
19	Point ZM	SMA 1 Makassar	2	2	2	2
3	Point ZM	Sungai Jeneberang	3	3	3	3
4	Point ZM	Sungai Jeneberang	3	3	3	3
5	Point ZM	Sungai Jeneberang	3	3	3	3
6	Point ZM	Sungai Pampang	3	3	1	3
7	Point ZM	Sungai Pampang	3	3	3	3
8	Point ZM	Sungai Pampang	3	3	2	2
9	Point ZM	Sungai Pampang	3	2	3	2
11	Point ZM	Sungai Tello	3	1	3	3
12	Point ZM	Sungai Tello	3	1	3	3
13	Point ZM	Sungai Tello	3	3	3	3

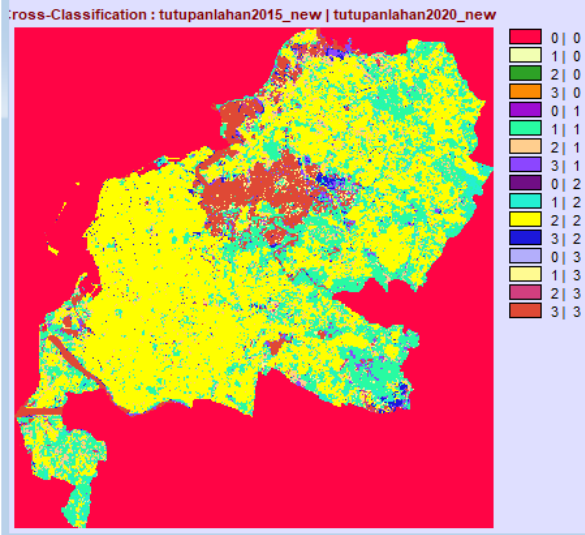
5. **Pengolahan Model Simulasi Prediksi Tutupan Lahan tahun 2025**

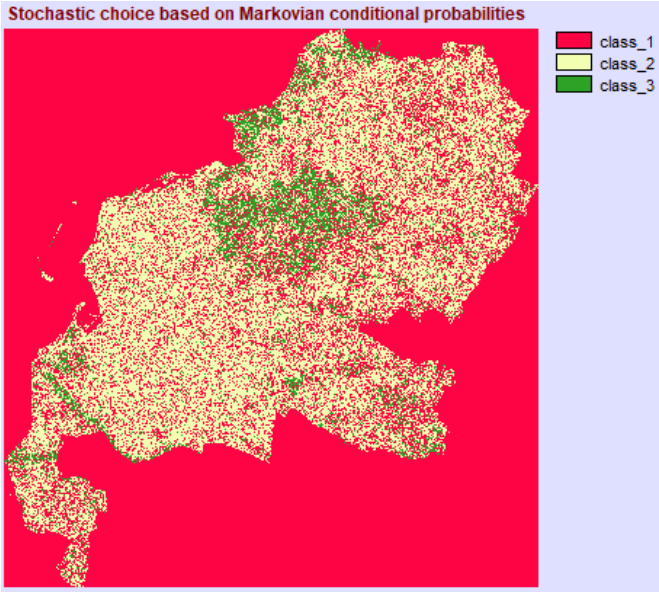



Tutupan Lahan Kota Makassar Tahun 2015
0
1
2
3

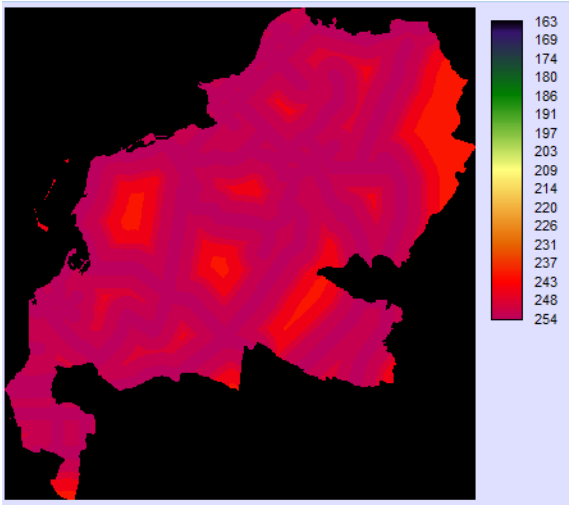

Tutupan Lahan Kota Makassar Tahun 2020
0
1
2
3


Cross-Classification : tutupanlahan2010_new | tutupanlahan2015_new
0 | 0
1 | 0
2 | 0
3 | 0
0 | 1
1 | 1
2 | 1
3 | 1
0 | 2
1 | 2
2 | 2
3 | 2
0 | 3
1 | 3
2 | 3
3 | 3

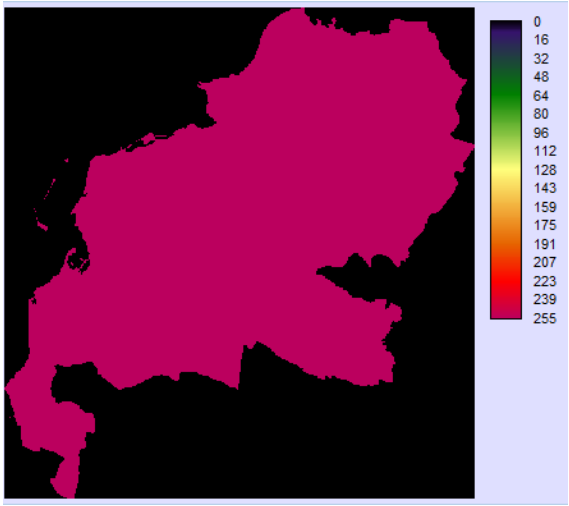

Cross-Classification : tutupanlahan2015_new | tutupanlahan2020_new
0 | 0
1 | 0
2 | 0
3 | 0
0 | 1
1 | 1
2 | 1
3 | 1
0 | 2
1 | 2
2 | 2
3 | 2
0 | 3
1 | 3
2 | 3
3 | 3


Stochastic choice based on Markovian conditional probabilities
class_1
class_2
class_3

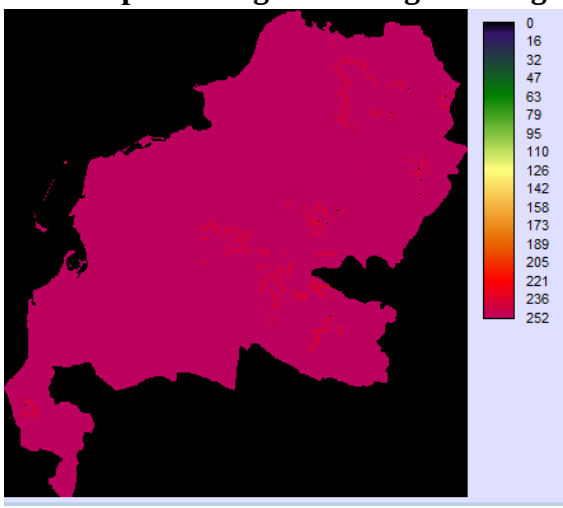
Faktor Pendorong : Jarak ke Jalan



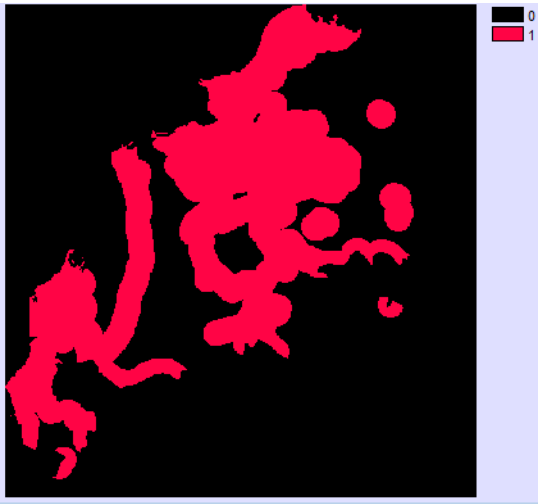
Faktor Pendorong : Jarak ke permukiman



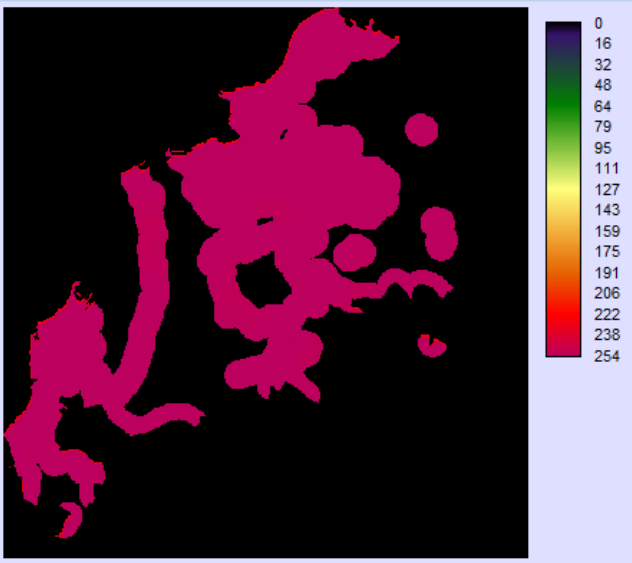
Faktor pendorong : kemiringan lereng



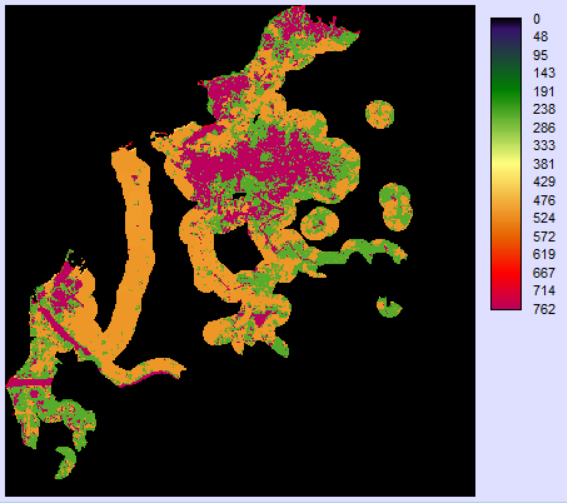
Faktor Pembatas : Tubuh air



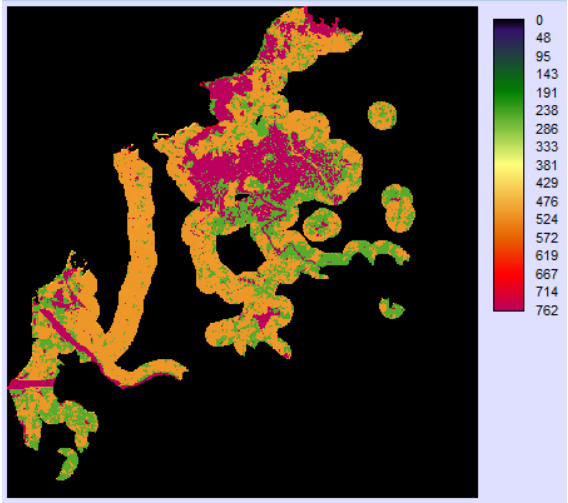
Hasil MCE



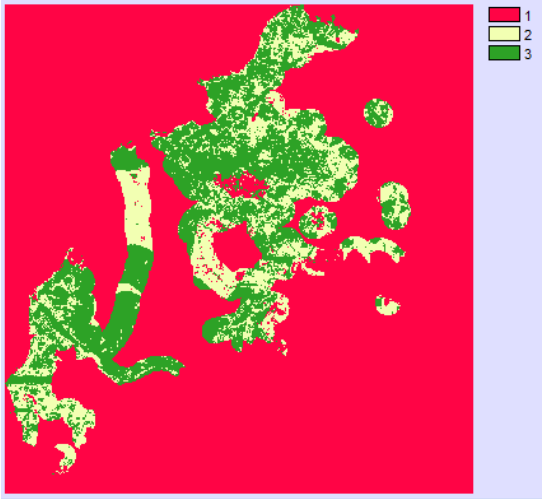
Hasil Overlay MCE- Tutupan Lahan 2010



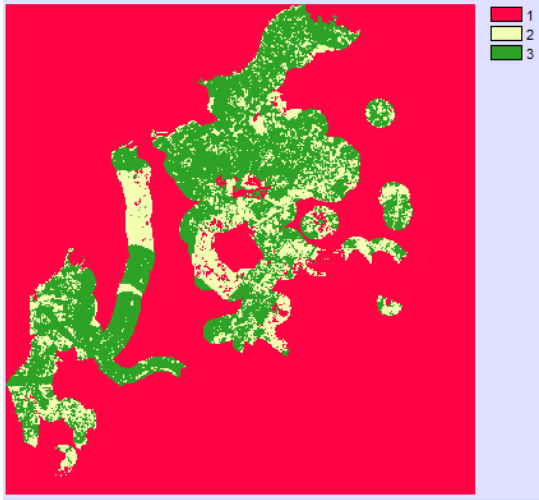
Hasil Overlay MCE-Tutupan Lahan 2015

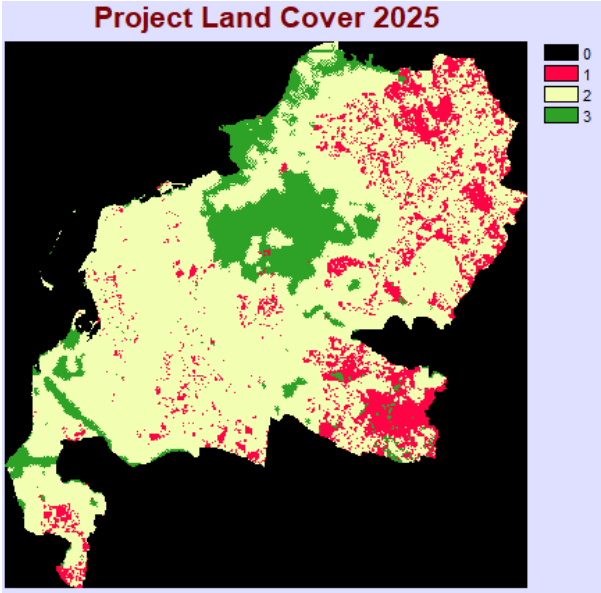


***Reclass* Overlay MCE-Tutupan Lahan 2010**



Reclass Overlay MCE-Tutupan Lahan 2015





LAMPIRAN 2 : Pengolahan Data LST

1. Script penentuan LST Tahun 2010

LST L7 single | Get Link | Save | Run | Reset | Apps

```
Imports (1 entry)
  var geometry: Polygon, 4 vertices
// link to the code that computes the Landsat LST
var LandsatLST = require('users/Mirnayani/GEE:Skripsi/Landsat_LST.js')


// select region of interest, date range, and landsat satellite
var satellite = 'L7';
var date_start = '2010-09-02';
var date_end = '2010-09-03';
var use_ndvi = true;

// get landsat collection with added variables: NDVI, FVC, TPW, EM, LST
var LandsatColl = LandsatLST.collection(satellite, date_start, date_end, geometry, use_ndvi)
  .filter(ee.Filter.eq('WRS_PATH', 114))
  .filter(ee.Filter.eq('WRS_ROW', 64))
  .filter(ee.Filter.lt('CLOUD_COVER',10))

print(LandsatColl)

// select the first feature
var exImage = LandsatColl.first();
print(exImage)

var fill_exImage= exImage.focal_mean(1, "square", "pixels", 20)
var final_image = fill_exImage.blend(exImage)

var cmap1 = ['blue', 'green', 'yellow', 'orange', 'red'];
var cmap2 = ['F2F2F2','EFC2B3','ECB176','E9BD3A','E6E600','63C600','00A600'];

Map.centerObject(geometry, 10)
Map.addLayer(final_image.select('TPW'),{min:0.0, max:60.0, palette:cmap1},'TCWV')
Map.addLayer(final_image.select('TPWpos'),{min:0.0, max:9.0, palette:cmap1},'TCWVpos')
Map.addLayer(final_image.select('FVC'),{min:0.0, max:1.0, palette:cmap2}, 'FVC')
Map.addLayer(final_image.select('EM'),{min:0.9, max:1.0, palette:cmap1}, 'Emissivity')
Map.addLayer(final_image.select('B6_VCID_2'),{min:290, max:320, palette:cmap1}, 'TIR BT')
Map.addLayer(final_image.select('LST'),{min:290, max:320, palette:cmap1}, 'LST')
Map.addLayer(final_image.select('B3', 'B2', 'B1'), {min:0, max:0.4}, 'RGB')


// uncomment the code below to export a image band to your drive
/*
Export.image.toDrive({
  image: final_image.select('LST'),
  description: 'Data_LST_2010',
  scale: 30,
  region: geometry,
  fileFormat: 'GeoTIFF',
});
*/

/*
Reference :
Ermida, S.L., Soares, P., Mantas, V., Göttsche, F.-M., Trigo, I.F., 2020.
    Google Earth Engine open-source code for Land Surface Temperature estimation from the Landsat series.
    Remote Sensing, 12 (9), 1471; https://doi.org/10.3390/rs12091471
*/
```

2. Script penentuan LST tahun 2015

LST L8 single * | Get Link | Save | Run | Reset | Apps

```
Imports (1 entry)
  var geometry: Polygon, 4 vertices
// link to the code that computes the Landsat LST
var LandsatLST = require('users/Mirnayani/GEE:Skripsi/Landsat_LST.js')


// select region of interest, date range, and landsat satellite
var satellite = 'L8';
var date_start = '2015-07-22';
var date_end = '2015-07-23';
var use_ndvi = true;

// get landsat collection with added variables: NDVI, FVC, TPW, EM, LST
var LandsatColl = LandsatLST.collection(satellite, date_start, date_end, geometry, use_ndvi)
  .filter(ee.Filter.eq('WRS_PATH', 114))
  .filter(ee.Filter.eq('WRS_ROW', 64))
  .filter(ee.Filter.lt('CLOUD_COVER',10))

print(LandsatColl)
```

```
// select the first feature
var exImage = LandsatColl.first();
print(exImage)

var cmap1 = ['blue', 'green', 'yellow', 'orange', 'red'];
var cmap2 = ['F2F2F2','EFC2B3','ECB176','E9BD3A','E6E600','63C600','00A600'];

Map.centerObject(geometry, 10)
Map.addLayer(exImage.select('TPW'),{min:0.0, max:60.0, palette:cmap1},'TCWV')
Map.addLayer(exImage.select('TPWpos'),{min:0.0, max:9.0, palette:cmap1},'TCWVpos')
Map.addLayer(exImage.select('FVC'),{min:0.0, max:1.0, palette:cmap2}, 'FVC')
Map.addLayer(exImage.select('EM'),{min:0.9, max:1.0, palette:cmap1}, 'Emissivity')
Map.addLayer(exImage.select('B10'),{min:290, max:320, palette:cmap1}, 'TIR BT')
Map.addLayer(exImage.select('LST'),{min:290, max:320, palette:cmap1}, 'LST')
Map.addLayer(exImage.select('B4', 'B3', 'B2'), {min:0, max:0.4}, 'RGB')
```


```
Map.addLayer(exImage.select('B4', 'B3', 'B2'), {min:0, max:0.4}, 'RGB')
```


```
// uncomment the code below to export a image band to your drive

/*
Export.image.toDrive({
  image: exImage.select('LST'),
  description: 'Data_LST_2015',
  scale: 30,
  region: geometry,
  fileFormat: 'GeoTIFF',
});
*/

/*
Reference :
Ermida, S.L., Soares, P., Mantas, V., Göttsche, F.-M., Trigo, I.F., 2020.
    Google Earth Engine open-source code for Land Surface Temperature estimation from the Landsat series.
    Remote Sensing, 12 (9), 1471; https://doi.org/10.3390/rs12091471
*/
```

3. Script penentuan LST tahun 2020

LST L8 single | Get Link | Save | Run | Reset | Apps

```
Imports (1 entry)
  var geometry: Polygon, 4 vertices
```

```
// link to the code that computes the Landsat LST
var LandsatLST = require('users/Mirnayani/GEE:Skripsi/Landsat_LST.js')


// select region of interest, date range, and landsat satellite
var satellite = 'L8';
var date_start = '2020-08-20';
var date_end = '2020-08-21';
var use_ndvi = true;

// get landsat collection with added variables: NDVI, FVC, TPW, EM, LST
var LandsatColl = LandsatLST.collection(satellite, date_start, date_end, geometry, use_ndvi)
  .filter(ee.Filter.eq('WRS_PATH', 114))
  .filter(ee.Filter.eq('WRS_ROW', 64))
  .filter(ee.Filter.lt('CLOUD_COVER',10))

print(LandsatColl)

// select the first feature
var exImage = LandsatColl.first();
print(exImage)

var cmap1 = ['blue', 'green', 'yellow', 'orange', 'red'];
var cmap2 = ['F2F2F2','EFC2B3','ECB176','E9BD3A','E6E600','63C600','00A600'];

Map.centerObject(geometry, 10)
Map.addLayer(exImage.select('TPW'),{min:0.0, max:60.0, palette:cmap1},'TCWV')
Map.addLayer(exImage.select('TPWpos'),{min:0.0, max:9.0, palette:cmap1},'TCWVpos')
Map.addLayer(exImage.select('FVC'),{min:0.0, max:1.0, palette:cmap2}, 'FVC')
Map.addLayer(exImage.select('EM'),{min:0.9, max:1.0, palette:cmap1}, 'Emissivity')
Map.addLayer(exImage.select('B10'),{min:290, max:320, palette:cmap1}, 'TIR BT')
Map.addLayer(exImage.select('LST'),{min:290, max:320, palette:cmap1}, 'LST')
Map.addLayer(exImage.select('B4', 'B3', 'B2'), {min:0, max:0.4}, 'RGB')

// uncomment the code below to export a image band to your drive

/*
Export.image.toDrive({
  image: exImage.select('LST'),
  description: 'Data_LST_2020',
  scale: 30,
  region: geometry,
  fileFormat: 'GeoTIFF',
});
*/

/*
```

```
Reference :
Ermida, S.L., Soares, P., Mantas, V., Göttsche, F.-M., Trigo, I.F., 2020.
    Google Earth Engine open-source code for Land Surface Temperature estimation from the Landsat series.
    Remote Sensing, 12 (9), 1471; https://doi.org/10.3390/rs12091471
*/
```

4. Input script penentuan LST berdasarkan penelitian Ermida et al., 2020

Landsat_LST.js Get Link Save Run Reset Apps

```
/*
Author: Sofia Ermida (sofia.ermida@ipma.pt; @ermida_sofia)

This code is free and open.
By using this code and any data derived with it,
you agree to cite the following reference
in any publications derived from them:
Ermida, S.L., Soares, P., Mantas, V., Göttsche, F.-M., Trigo, I.F., 2020.
    Google Earth Engine open-source code for Land Surface Temperature estimation from the Landsat series.
    Remote Sensing, 12 (9), 1471; https://doi.org/10.3390/rs12091471

This function selects the Landsat data based on user inputs
and performes the LST computation

to call this function use:

var LandsatLST = require('users/sofiaermida/landsat_smw_lst:modules/Landsat_LST.js')
var LandsatCollection = LandsatLST.collection(landsat, date_start, date_end, geometry)

USES:
    - NCEP_TPW.js
    - cloudmask.js
    - compute_NDVI.js
    - compute_FVC.js
    - compute_emissivity.js
    - SMWalgorithm.js

INPUTS:
        - landsat: <string>
                the Landsat satellite id
                valid inputs: 'L4', 'L5', 'L7' and 'L8'
        - date_start: <string>
                start date of the Landsat collection
                format: YYYY-MM-DD
        - date_end: <string>
                end date of the Landsat collection
                format: YYYY-MM-DD
        - geometry: <ee.Geometry>
                region of interest
        - use_ndvi: <boolean>
                if true, NDVI values are used to obtain a
                dynamic emissivity; if false, emissivity is
                obtained directly from ASTER
OUTPUTS:
        - <ee.ImageCollection>
          image collection with bands:
          - landsat original bands: all from SR excpet the TIR bands (from TOA)
          - cloud masked
          - 'NDVI': normalized vegetation index
          - 'FVC': fraction of vegetation cover [0-1]
          - 'TPW': total precipitable water [mm]
          - 'EM': surface emissvity for TIR band
          - 'LST': land surface temperature

  14-08-2020: update to avoid using the getInfo() and if()
    (Thanks Tyler Erickson for the suggestion)
*/

// MODULES DECLARATION ----------------------------------------------------------
// Total Precipitable Water
var NCEP_TPW = require('users/sofiaermida/landsat_smw_lst:modules/NCEP_TPW.js')
//cloud mask
var cloudmask = require('users/sofiaermida/landsat_smw_lst:modules/cloudmask.js')
//Normalized Difference Vegetation Index
var NDVI = require('users/sofiaermida/landsat_smw_lst:modules/compute_NDVI.js')
//Fraction of Vegetation cover
var FVC = require('users/sofiaermida/landsat_smw_lst:modules/compute_FVC.js')
//surface emissivity
var EM = require('users/sofiaermida/landsat_smw_lst:modules/compute_emissivity.js')
// land surface temperature
var LST = require('users/sofiaermida/landsat_smw_lst:modules/SMWalgorithm.js')
// ------------------------------------------------------------------------------


var COLLECTION = ee.Dictionary({
  'L7': {
    'TOA': ee.ImageCollection('LANDSAT/LE07/C01/T1_TOA'),
    'TIR': ['B6_VCID_1','B6_VCID_2'],
  },
  'L8': {
```

```
    'TOA': ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA'),
    'TIR': ['B10','B11']
  }
});


exports.collection = function(landsat, date_start, date_end, geometry, use_ndvi) {

  // load TOA Radiance/Reflectance
  var collection_dict = ee.Dictionary(COLLECTION.get(landsat));

  var landsatTOA = ee.ImageCollection(collection_dict.get('TOA'))
                .filter(ee.Filter.date(date_start, date_end))
                .filterBounds(geometry)
                .map(cloudmask.toa)
                .map(NDVI.addBand(landsat))
                .map(FVC.addBand(landsat))
                .map(NCEP_TPW.addBand)
                .map(EM.addBand(landsat,use_ndvi));

  // compute the LST
  var tir = ee.List(collection_dict.get('TIR'))
  var landsatLST = landsatTOA.map(LST.addBand(landsat));

  return landsatLST;
};
```

NCEP_TPW.js

Get Link | Save | Run | Reset | Apps

```
/*
Author: Sofia Ermida (sofia.ermida@ipma.pt; @ermida_sofia)

This code is free and open.
By using this code and any data derived with it,
you agree to cite the following reference
in any publications derived from them:
Ermida, S.L., Soares, P., Mantas, V., Göttsche, F.-M., Trigo, I.F., 2020.
    Google Earth Engine open-source code for Land Surface Temperature estimation from the Landsat series.
    Remote Sensing, 12 (9), 1471; https://doi.org/10.3390/rs12091471

this function matches the atmospheric water vapour data
from NCEP reanalysis to each Landsat image
tpw values are interpolated from the 6-hourly model times to the image time

to call this function use:

var NCEP_TPW = require('users/sofiaermida/landsat_smw_lst:modules/NCEP_TPW.js')
var ImagewithTPW = NCEP_TPW.addBand(image)
or
var collectionwithPTW = ImageCollection.map(NCEP_TPW.addBand)

INPUTS:
        - image: <ee.Image>
                image for which to interpolate the TPW data
                needs the 'system:time_start' image property
OUTPUTS:
        - <ee.Image>
          the input image with 3 new bands:
          'TPW': total precipitable water values
          'TPWpos': index for the LUT of SMW algorithm coefficients

  10.12.2020: typo correction in the tpw inperpolation expression
    (thanks to Jiacheng Zhao for reporting this issue)
*/

exports.addBand = function(image){

  // first select the day of interest
  var date = ee.Date(image.get('system:time_start'))
  var year = ee.Number.parse(date.format('yyyy'))
  var month = ee.Number.parse(date.format('MM'))
  var day = ee.Number.parse(date.format('dd'))
  var date1 = ee.Date.fromYMD(year,month,day)
  var date2 = date1.advance(1,'days')

  // function compute the time difference from landsat image
  var datedist = function(image){
    return image.set('DateDist',
      ee.Number(image.get('system:time_start'))
      .subtract(date.millis()).abs())
  };

  // load atmospheric data collection
  var TPWcollection = ee.ImageCollection('NCEP_RE/surface_wv')
                .filter(ee.Filter.date(date1.format('yyyy-MM-dd'), date2.format('yyyy-MM-dd')))
                .map(datedist)

  // select the two closest model times
    var closest = (TPWcollection.sort('DateDist')).toList(2);
```

```
61
62    // check if there is atmospheric data in the wanted day
63    // if not creates a TPW image with non-realistic values
64    // these are then masked in the SMWalgorithm function (prevents errors)
65    var tpw1 = ee.Image(ee.Algorithms.If(closest.size().eq(0), ee.Image.constant(-999.0),
66                        ee.Image(closest.get(0)).select('pr_wtr') ));
67    var tpw2 = ee.Image(ee.Algorithms.If(closest.size().eq(0), ee.Image.constant(-999.0),
68                        ee.Algorithms.If(closest.size().eq(1), tpw1,
69                        ee.Image(closest.get(1)).select('pr_wtr') )));
70
71    var time1 = ee.Number(ee.Algorithms.If(closest.size().eq(0), 1.0,
72                        ee.Number(tpw1.get('DateDist')).divide(ee.Number(21600000)) ));
73    var time2 = ee.Number(ee.Algorithms.If(closest.size().lt(2), 0.0,
74                        ee.Number(tpw2.get('DateDist')).divide(ee.Number(21600000)) ));
75
76    var tpw = tpw1.expression('tpw1*time2+tpw2*time1',
77                        {'tpw1':tpw1,
78                        'time1':time1,
79                        'tpw2':tpw2,
80                        'time2':time2
81                        }).clip(image.geometry());
82
83    // SMW coefficients are binned by TPW values
84    // find the bin of each TPW value
85    var pos = tpw.expression(
86      "value = (TPW>0 && TPW<=6) ? 0" +
87      ": (TPW>6 && TPW<=12) ? 1" +
88      ": (TPW>12 && TPW<=18) ? 2" +
89      ": (TPW>18 && TPW<=24) ? 3" +
90      ": (TPW>24 && TPW<=30) ? 4" +
91      ": (TPW>30 && TPW<=36) ? 5" +
92      ": (TPW>36 && TPW<=42) ? 6" +
93      ": (TPW>42 && TPW<=48) ? 7" +
94      ": (TPW>48 && TPW<=54) ? 8" +
95      ": (TPW>54) ? 9" +
96      ": 0",{'TPW': tpw})
97      .clip(image.geometry());
98
99    // add tpw to image as a band
100   var withTPW = (image.addBands(tpw.rename('TPW'),['TPW'])).addBands(pos.rename('TPWpos'),['TPWpos']);
101
102   return withTPW
103 };
```

SMW_coefficients.js

Get Link | Save | Run | Reset | Apps

```
1   /*
2   Author: Sofia Ermida (sofia.ermida@ipma.pt; @ermida_sofia)
3
4   This code is free and open.
5   By using this code and any data derived with it,
6   you agree to cite the following reference
7   in any publications derived from them:
8   Ermida, S.L., Soares, P., Mantas, V., Göttsche, F.-M., Trigo, I.F., 2020.
9       Google Earth Engine open-source code for Land Surface Temperature estimation from the Landsat series.
10      Remote Sensing, 12 (9), 1471; https://doi.org/10.3390/rs12091471
11  */
12  // coefficients for the Statistical Mono-Window Algorithm
13  exports.coeff_SMW_L4 = ee.FeatureCollection([
14    ee.Feature(null, {'TPWpos': 0, 'A': 0.9755, 'B': -205.2767, 'C': 212.0051}),
15    ee.Feature(null, {'TPWpos': 1, 'A': 1.0155, 'B': -233.8902, 'C': 230.4049}),
16    ee.Feature(null, {'TPWpos': 2, 'A': 1.0672, 'B': -257.1884, 'C': 239.3072}),
17    ee.Feature(null, {'TPWpos': 3, 'A': 1.1499, 'B': -286.2166, 'C': 244.8497}),
18    ee.Feature(null, {'TPWpos': 4, 'A': 1.2277, 'B': -316.7643, 'C': 253.0033}),
19    ee.Feature(null, {'TPWpos': 5, 'A': 1.3649, 'B': -361.8276, 'C': 258.5471}),
20    ee.Feature(null, {'TPWpos': 6, 'A': 1.5085, 'B': -410.1157, 'C': 265.1131}),
21    ee.Feature(null, {'TPWpos': 7, 'A': 1.7045, 'B': -472.4909, 'C': 270.7000}),
22    ee.Feature(null, {'TPWpos': 8, 'A': 1.5886, 'B': -442.9489, 'C': 277.1511}),
23    ee.Feature(null, {'TPWpos': 9, 'A': 2.0215, 'B': -571.8563, 'C': 279.9854})
24  ]);
25
26  exports.coeff_SMW_L5 = ee.FeatureCollection([
27    ee.Feature(null, {'TPWpos': 0, 'A': 0.9765, 'B': -204.6584, 'C': 211.1321}),
28    ee.Feature(null, {'TPWpos': 1, 'A': 1.0229, 'B': -235.5384, 'C': 230.0619}),
29    ee.Feature(null, {'TPWpos': 2, 'A': 1.0817, 'B': -261.3886, 'C': 239.5256}),
30    ee.Feature(null, {'TPWpos': 3, 'A': 1.1738, 'B': -293.6128, 'C': 245.6042}),
31    ee.Feature(null, {'TPWpos': 4, 'A': 1.2605, 'B': -327.1417, 'C': 254.2301}),
32    ee.Feature(null, {'TPWpos': 5, 'A': 1.4166, 'B': -377.7741, 'C': 259.9711}),
33    ee.Feature(null, {'TPWpos': 6, 'A': 1.5727, 'B': -430.0388, 'C': 266.9520}),
34    ee.Feature(null, {'TPWpos': 7, 'A': 1.7879, 'B': -498.1947, 'C': 272.8413}),
35    ee.Feature(null, {'TPWpos': 8, 'A': 1.6347, 'B': -457.8183, 'C': 279.6160}),
36    ee.Feature(null, {'TPWpos': 9, 'A': 2.1168, 'B': -600.7079, 'C': 282.4583})
37  ]);
38
39  exports.coeff_SMW_L7 = ee.FeatureCollection([
40    ee.Feature(null, {'TPWpos': 0, 'A': 0.9764, 'B': -205.3511, 'C': 211.8507}),
41    ee.Feature(null, {'TPWpos': 1, 'A': 1.0201, 'B': -235.2416, 'C': 230.5468}),
42    ee.Feature(null, {'TPWpos': 2, 'A': 1.0750, 'B': -259.6560, 'C': 239.6619}),
43    ee.Feature(null, {'TPWpos': 3, 'A': 1.1612, 'B': -289.8190, 'C': 245.3286}),
44    ee.Feature(null, {'TPWpos': 4, 'A': 1.2425, 'B': -321.4658, 'C': 253.6144}),
```

```
  ee.Feature(null, {'TPWpos': 5, 'A': 1.3864, 'B': -368.4078, 'C': 259.1390}),
  ee.Feature(null, {'TPWpos': 6, 'A': 1.5336, 'B': -417.7796, 'C': 265.7486}),
  ee.Feature(null, {'TPWpos': 7, 'A': 1.7345, 'B': -481.5714, 'C': 271.3659}),
  ee.Feature(null, {'TPWpos': 8, 'A': 1.6066, 'B': -448.5071, 'C': 277.9058}),
  ee.Feature(null, {'TPWpos': 9, 'A': 2.0533, 'B': -581.2619, 'C': 280.6800})
]);

exports.coeff_SMW_L8 = ee.FeatureCollection([
  ee.Feature(null, {'TPWpos': 0, 'A': 0.9751, 'B': -205.8929, 'C': 212.7173}),
  ee.Feature(null, {'TPWpos': 1, 'A': 1.0090, 'B': -232.2750, 'C': 230.5698}),
  ee.Feature(null, {'TPWpos': 2, 'A': 1.0541, 'B': -253.1943, 'C': 238.9548}),
  ee.Feature(null, {'TPWpos': 3, 'A': 1.1282, 'B': -279.4212, 'C': 244.0772}),
  ee.Feature(null, {'TPWpos': 4, 'A': 1.1987, 'B': -307.4497, 'C': 251.8341}),
  ee.Feature(null, {'TPWpos': 5, 'A': 1.3205, 'B': -348.0228, 'C': 257.2740}),
  ee.Feature(null, {'TPWpos': 6, 'A': 1.4540, 'B': -393.1718, 'C': 263.5599}),
  ee.Feature(null, {'TPWpos': 7, 'A': 1.6350, 'B': -451.0790, 'C': 268.9405}),
  ee.Feature(null, {'TPWpos': 8, 'A': 1.5468, 'B': -429.5095, 'C': 275.0895}),
  ee.Feature(null, {'TPWpos': 9, 'A': 1.9403, 'B': -547.2681, 'C': 277.9953})
]);
```

SMWalgorithm.js Get Link Save Run Reset Apps

```
/*
Author: Sofia Ermida (sofia.ermida@ipma.pt; @ermida_sofia)

This code is free and open.
By using this code and any data derived with it,
you agree to cite the following reference
in any publications derived from them:
Ermida, S.L., Soares, P., Mantas, V., Göttsche, F.-M., Trigo, I.F., 2020.
    Google Earth Engine open-source code for Land Surface Temperature estimation from the Landsat series.
    Remote Sensing, 12 (9), 1471; https://doi.org/10.3390/rs12091471

This function applies the Stastical Mono-Window algorithm to compute the LST

to call this function use:

var LSTfun = require('users/sofiaermida/landsat_smw_lst:modules/SMWalgorithm.js')
var ImagewithLST = LSTfun.addBand(landsat)(image)
or
var collectionwithLST = ImageCollection.map(LSTfun.addBand(landsat))

USES:
    - SMW_coefficients.js

INPUTS:
        - landsat: <string>
                the Landsat satellite id
                valid inputs: 'L4', 'L5', 'L7' and 'L8'
        - image: <ee.Image>
                image for which to calculate the LSTy
OUTPUTS:
        - <ee.Image>
          the input image with 1 new band:
          'LST': land surface temperature
*/


// coefficients for the Statistical Mono-Window Algorithm
var SMWcoef = require('users/sofiaermida/landsat_smw_lst:modules/SMW_coefficients.js');

// Function to create a lookup between two columns in a
// feature collection
var get_lookup_table = function(fc, prop_1, prop_2) {
  var reducer = ee.Reducer.toList().repeat(2);
  var lookup = fc.reduceColumns(reducer, [prop_1, prop_2]);
  return ee.List(lookup.get('list'));
};


exports.addBand = function(landsat){

  var wrap = function(image){

    // Select algorithm coefficients
    var coeff_SMW = ee.FeatureCollection(ee.Algorithms.If(landsat=='L4',SMWcoef.coeff_SMW_L4,
                                         ee.Algorithms.If(landsat=='L5',SMWcoef.coeff_SMW_L5,
                                         ee.Algorithms.If(landsat=='L7',SMWcoef.coeff_SMW_L7,
                                         SMWcoef.coeff_SMW_L8))));

    // Create lookups for the algorithm coefficients
    var A_lookup = get_lookup_table(coeff_SMW, 'TPWpos', 'A');
    var B_lookup = get_lookup_table(coeff_SMW, 'TPWpos', 'B');
    var C_lookup = get_lookup_table(coeff_SMW, 'TPWpos', 'C');

    // Map coefficients to the image using the TPW bin position
    var A_img = image.remap(A_lookup.get(0), A_lookup.get(1),0.0,'TPWpos').resample('bilinear');
    var B_img = image.remap(B_lookup.get(0), B_lookup.get(1),0.0,'TPWpos').resample('bilinear');
    var C_img = image.remap(C_lookup.get(0), C_lookup.get(1),0.0,'TPWpos').resample('bilinear');

    // select TIR band
    var tir = ee.String(ee.Algorithms.If(landsat=='L8','B10',
```

```
                    ee.Algorithms.If(landsat=='L7','B6_VCID_1',
                    'B6')));
    // compute the LST
    var lst = image.expression(
      'A*Tb1/em1 + B/em1 + C',
        {'A': A_img,
         'B': B_img,
         'C': C_img,
         'em1': image.select('EM'),
         'Tb1': image.select(tir)
        }).updateMask(image.select('TPW').lt(0).not());


    return image.addBands(lst.rename('LST'))
  };
  return wrap
}
```

compute_FVC.js

Get Link | Save | Run | Reset | Apps

```
to call this function use:

var FVCfun = require('users/sofiaermida/landsat_smw_lst:modules/compute_FVC.js')
var ImagewithFVC = FVCfun.addBand(landsat)(image)
or
var collectionwithFVC = ImageCollection.map(FVCfun.addBand(landsat))

USES:
    - SMW_coefficients.js

INPUTS:
        - landsat: <string>
                the Landsat satellite id
                valid inputs: 'L4', 'L5', 'L7' and 'L8'
        - image: <ee.Image>
                image for which to calculate the FVC
OUTPUTS:
        - <ee.Image>
          the input image with 1 new band:
          'FVC': fraction of vegetation cover
*/
exports.addBand = function(landsat){
  var wrap = function(image){

    var ndvi = image.select('NDVI')

    // Compute FVC
    var fvc = image.expression('((ndvi-ndvi_bg)/(ndvi_vg - ndvi_bg))**2',
      {'ndvi':ndvi,'ndvi_bg':0.2,'ndvi_vg':0.86});
    fvc = fvc.where(fvc.lt(0.0),0.0);
    fvc = fvc.where(fvc.gt(1.0),1.0);

    return image.addBands(fvc.rename('FVC'));
  }
  return wrap
};
```

compute_NDVI.js

Get Link | Save | Run | Reset | Apps

```


to call this function use:

var NDVIfun = require('users/sofiaermida/landsat_smw_lst:modules/compute_NDVI.js')
var ImagewithNDVI = NDVIfun.addBand(landsat)(image)
or
var collectionwithNDVI = ImageCollection.map(NDVIfun.addBand(landsat))

INPUTS:
        - landsat: <string>
                the Landsat satellite id
                valid inputs: 'L4', 'L5', 'L7' and 'L8'
        - image: <ee.Image>
                image for which to calculate the NDVI
OUTPUTS:
        - <ee.Image>
          the input image with 1 new band:
          'NDVI': normalized difference vegetation index
*/

exports.addBand = function(landsat){
  var wrap = function(image){

    // choose bands
    var nir = ee.String(ee.Algorithms.If(landsat=='L8','B5','B4'))
    var red = ee.String(ee.Algorithms.If(landsat=='L8','B4','B3'))

    // compute NDVI
    return image.addBands(image.expression('(nir-red)/(nir+red)',{
      'nir':image.select(nir).multiply(0.0001),
      'red':image.select(red).multiply(0.0001)
```

```
    }).rename('NDVI'))
  }
  return wrap
};
```

compute_emissivity.js

```
        'EM': surface emissivity of TIR band
*/

var ASTERGED = require('users/sofiaermida/landsat_smw_lst:modules/ASTER_bare_emiss.js')

// this function computes the emissivity of the
// Landsat TIR band using ASTER and FVC
exports.addBand = function(landsat, use_ndvi){
  var wrap = function(image){

    var c13 = ee.Number(ee.Algorithms.If(landsat=='L4',0.3222,
                        ee.Algorithms.If(landsat=='L5',-0.0723,
                        ee.Algorithms.If(landsat=='L7',0.2147,
                        0.6820))));
    var c14 = ee.Number(ee.Algorithms.If(landsat=='L4',0.6498,
                        ee.Algorithms.If(landsat=='L5',1.0521,
                        ee.Algorithms.If(landsat=='L7',0.7789,
                        0.2578))));
    var c = ee.Number(ee.Algorithms.If(landsat=='L4',0.0272,
                        ee.Algorithms.If(landsat=='L5',0.0195,
                        ee.Algorithms.If(landsat=='L7',0.0059,
                        0.0584))));

    // get ASTER emissivity
    // convolve to Landsat band
    var emiss_bare = image.expression('c13*EM13 + c14*EM14 + c',{
      'EM13':ASTERGED.emiss_bare_band13(image),
      'EM14':ASTERGED.emiss_bare_band14(image),
      'c13':ee.Image(c13),
      'c14':ee.Image(c14),
      'c':ee.Image(c)
      });

    // compute the dynamic emissivity for Landsat
    var EMd = image.expression('fvc*0.99+(1-fvc)*em_bare',
      {'fvc':image.select('FVC'),'em_bare':emiss_bare});

    // compute emissivity directly from ASTER
    // without vegetation correction
    // get ASTER emissivity
    var aster = ee.Image("NASA/ASTER_GED/AG100_003")
      .clip(image.geometry());
    var EM0 = image.expression('c13*EM13 + c14*EM14 + c',{
      'EM13':aster.select('emissivity_band13').multiply(0.001),
      'EM14':aster.select('emissivity_band14').multiply(0.001),
      'c13':ee.Image(c13),
      'c14':ee.Image(c14),
      'c':ee.Image(c)
      });

    // select which emissivity to output based on user selection
    var EM = ee.Image(ee.Algorithms.If(use_ndvi,EMd,EM0));

    return image.addBands(EM.rename('EM'));
  }
  return wrap
}
```

5. Script pengolahan NDBI, NDVI, NDWI tahun 2010

Predictor LST L7

```
Imports (1 entry)
  var geometry: Table users/Mirnayani/shpmakassar
//load Landsat 7 Tier 1 TOA image collection
var MIN_SCALE = 1/3;
var MAX_SCALE = 3;
var MIN_NEIGHBORS = 144;

/* Apply the USGS L7 Phase-2 Gap filling protocol, using a single kernel size. */
var GapFill = function(src, fill, kernelSize, upscale) {
  var kernel = ee.Kernel.square(kernelSize * 30, "meters", false)

  // Find the pixels common to both scenes.
  var common = src.mask().and(fill.mask())
  var fc = fill.updateMask(common)
  var sc = src.updateMask(common)
  Map.addLayer(common.select(0).mask(common.select(0)), {palette:['000000']}, 'common mask (both exist)', false)

  // Find the primary scaling factors with a regression.
  // Interleave the bands for the regression.  This assumes the bands have the same names.
  var regress = fc.addBands(sc)

  regress = regress.select(regress.bandNames().sort())

```

```
var ratio = 5

if(upscale) {
  var fit = regress
    .reduceResolution(ee.Reducer.median(), false, 500)
    .reproject(regress.select(0).projection().scale(ratio, ratio))
    .reduceNeighborhood(ee.Reducer.linearFit().forEach(src.bandNames()), kernel, null, false)
    .unmask()
    .reproject(regress.select(0).projection().scale(ratio, ratio))
} else {
  var fit = regress
    .reduceNeighborhood(ee.Reducer.linearFit().forEach(src.bandNames()), kernel, null, false)
}

var offset = fit.select(".*_offset")
var scale = fit.select(".*_scale")

Map.addLayer(scale.select('B1_scale'), {min:-2, max:2}, 'scale B1', false)

// Find the secondary scaling factors using just means and stddev
var reducer = ee.Reducer.mean().combine(ee.Reducer.stdDev(), null, true)

if(upscale) {
  var src_stats = src
    .reduceResolution(ee.Reducer.median(), false, 500)
    .reproject(regress.select(0).projection().scale(ratio, ratio))
    .reduceNeighborhood(reducer, kernel, null, false)
    .reproject(regress.select(0).projection().scale(ratio, ratio))

  var fill_stats = fill
    .reduceResolution(ee.Reducer.median(), false, 500)
    .reproject(regress.select(0).projection().scale(ratio, ratio))
    .reduceNeighborhood(reducer, kernel, null, false)
    .reproject(regress.select(0).projection().scale(ratio, ratio))
} else {
  var src_stats = src
    .reduceNeighborhood(reducer, kernel, null, false)

  var fill_stats = fill
    .reduceNeighborhood(reducer, kernel, null, false)
}

var scale2 = src_stats.select(".*stdDev").divide(fill_stats.select(".*stdDev"))
var offset2 = src_stats.select(".*mean").subtract(fill_stats.select(".*mean").multiply(scale2))

var invalid = scale.lt(MIN_SCALE).or(scale.gt(MAX_SCALE))
Map.addLayer(invalid.select(0).mask(invalid.select(0)), {palette:['550000']}, 'invalid1', false)
```

```
  scale = scale.where(invalid, scale2)
  offset = offset.where(invalid, offset2)

  // When all else fails, just use the difference of means as an offset.
  var invalid2 = scale.lt(MIN_SCALE).or(scale.gt(MAX_SCALE))
  Map.addLayer(invalid2.select(0).mask(invalid2.select(0)), {palette:['552020']}, 'invalid2', false)
  scale = scale.where(invalid2, 1)
  offset = offset.where(invalid2, src_stats.select(".*mean").subtract(fill_stats.select(".*mean")))

  // Apply the scaling and mask off pixels that didn't have enough neighbors.
  var count = common.reduceNeighborhood(ee.Reducer.count(), kernel, null, true, "boxcar")
  var scaled = fill.multiply(scale).add(offset)
      .updateMask(count.gte(MIN_NEIGHBORS))

  return src.unmask(scaled, true)
}

var source = ee.Image("LANDSAT/LE07/C01/T1_TOA/LE07_114064_20100902")
var fill = ee.Image("LANDSAT/LE07/C01/T1_TOA/LE07_114064_20100310")

/*
Map.addLayer(source, {bands: ['B3', 'B2', 'B1'], min: 0, max: 0.3}, 'Makassar_2020src');
Map.addLayer(fill, {bands: ['B3', 'B2', 'B1'], min: 0, max: 0.3}, 'Makassar_2020fill');
*/

/*
Map.addLayer(fill, {min:0, max:0.3, bands:["B3", "B2", "B1"]}, "fill", true)
Map.addLayer(source, {min:0, max:0.3, bands:["B3", "B2", "B1"]}, "destination", true)
*/

var L7 = GapFill(source, fill, 10, false);
Map.addLayer(L7, {min:0, max:0.3, bands:["B3", "B2", "B1"]}, "filled", true)

var L7 = GapFill(source, fill, 10, true);
Map.addLayer(L7, {min:0, max:0.3, bands:["B3", "B2", "B1"]}, "filled (upscaled)", true)



//Display the cliped image collection with visual parameters
Map.addLayer(L7, {bands: ['B3', 'B2', 'B1'], min: 0, max: 0.3}, 'Makassar_2010');
```

```
// Compute the Normalized Difference Vegetation Index (NDVI).
var addNDVI = function(L7) {
  var ndvi = L7.normalizedDifference(['B4', 'B3']).rename('NDVI');
  return L7.addBands(ndvi);
};

// Test the addNDVI function on a single image.
var ndvi = addNDVI(L7).select('NDVI');

// Display the result.
Map.centerObject(geometry, 9);
var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};
Map.addLayer(ndvi, ndviParams, 'NDVI image');

// Compute the Normalized Difference Water Index (NDWI).
var addNDWI = function(L7) {
  var ndwi = L7.normalizedDifference(['B4', 'B5']).rename('NDWI');
  return L7.addBands(ndwi);
};

// Test the addNDWI function on a single image.
var ndwi = addNDWI(L7).select('NDWI');

// Display the result.
Map.centerObject(geometry, 9);
var ndwiParams = {min: 0.5, max: 1, palette: ['00FFFF', '0000FF']};
Map.addLayer(ndwi, ndwiParams, 'NDWI image');

// Compute the Normalized Difference Built-up Index (NDBI).
var addNDBI = function(L7) {
  var ndbi = L7.normalizedDifference(['B5','B4']).rename('NDBI');
  return L7.addBands(ndbi);
};

// Test the addNDBI function on a single image.
var ndbi = addNDBI(L7).select('NDBI');

// Display the result.
Map.centerObject(geometry, 9);
var ndbiParams = {min: -1, max: 1, palette: ['white', 'orange', 'red']};
Map.addLayer(ndbi, ndbiParams, 'NDBI image');

var stacked_L7 = L7.addBands(ndvi).addBands(ndwi).addBands(ndbi);
print('stacked_L7', stacked_L7.bandNames());

//Display the stacked_composite
Map.addLayer(stacked_L7, {bands: ['B3', 'B2', 'B1'], min: 0, max: 0.3}, 'Makassar2010_stack');

/*
Export.image.toDrive({
  image: stacked_L7.select('NDVI'),
  description: 'Data_2010_NDVI',
  scale: 30,
  region: geometry,
  fileFormat: 'GeoTIFF',
});

Export.image.toDrive({
  image: stacked_L7.select('NDBI'),
  description: 'Data_2010_NDBI',
  scale: 30,
  region: geometry,
  fileFormat: 'GeoTIFF',
});

Export.image.toDrive({
  image: stacked_L7.select('NDWI'),
  description: 'Data_2010_NDWI',
  scale: 30,
  region: geometry,
  fileFormat: 'GeoTIFF',
});
*/
```

6. Data Suhu Udara Harian dari BMKG

ID WMO	:	97182
Nama Stasiun	:	Stasiun Meteorologi Maritim Paotere
Lintang	:	-5.11375
Bujur	:	119.41983
Elevasi	:	5

Tanggal	Tn	Tx	Tavg
02-09-2010	24,8	32,8	28,5
22-07-2015	23,8	32,2	27
20-08-2020	23,5	32	27,4

Keterangan :
Tn: Temperatur minimum (°C)
Tx: Temperatur maksimum (°C)
Tavg: Temperatur rata-rata (°C)

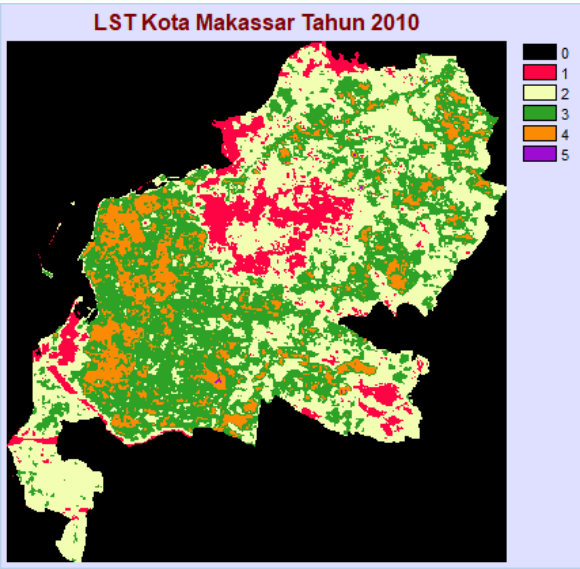
7. Data Suhu Udara Tahunan dari BMKG Wilayah IV Makassar

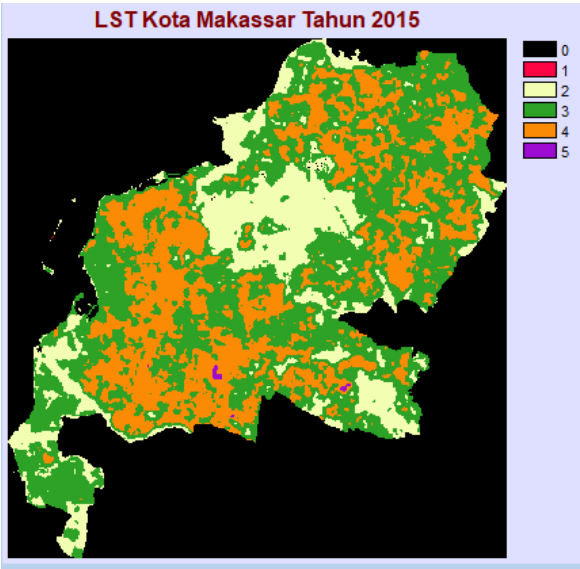
DATA SUHU UDARA
STASIUN METEOROLOGI MARITIM PAOTERE, KOTA MAKASSAR, SULAWESI SELATAN

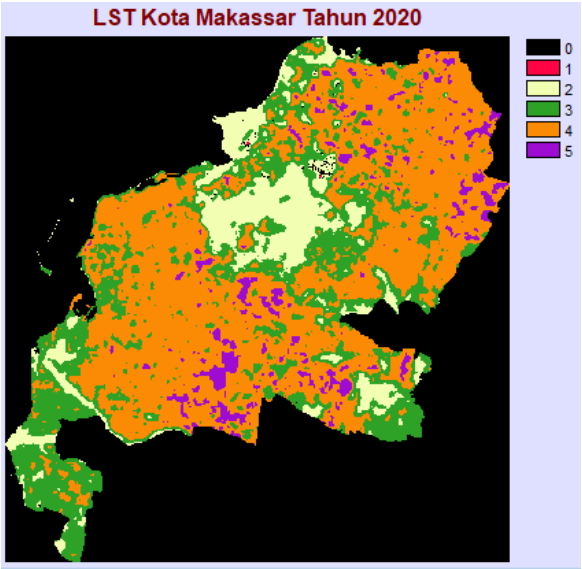
KOORDINAT : 05° 06' 49.5" LS, 119° 25' 11.4" BT
TINGGI : 1.75 m
SUHU UDARA RATA RATA (dalam derajat celcius)

BULAN / TAHUN	JAN	FEB	MAR	APR	MEI	JUN	JUL	AGS	SEP	OKT	NOV	DES
2010	26.6	27.8	28.2	28.5	28.5	28.1	27.8	28.1	28.0	28.1	28.2	26.6
2015	26.8	27.2	27.5	27.8	28.3	27.7	27.4	27.3	27.6	29.0	29.4	27.9
2020	28.3	27.8	28.3	28.5	29.0	28.4	28.2	28.6	29.1	29.3	28.9	27.0

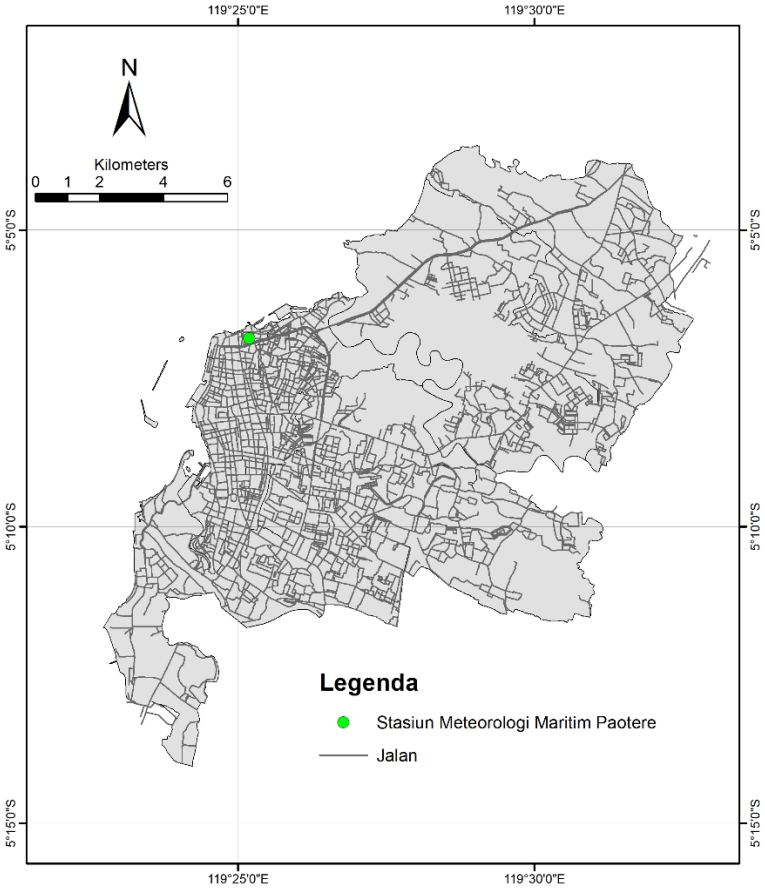
8. Pengolahan Model Simulasi Prediksi LST tahun 2025

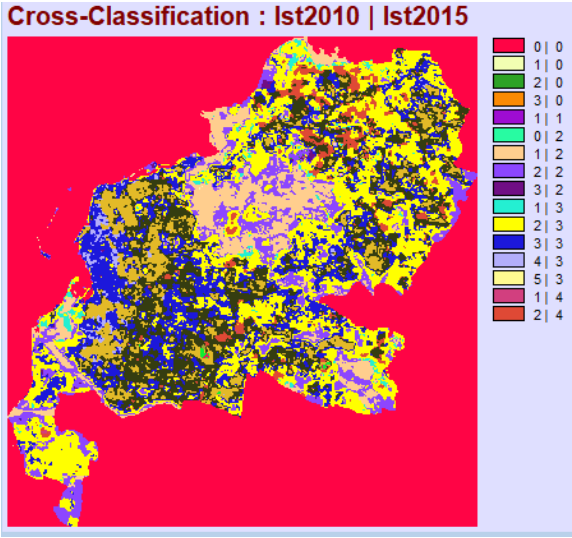


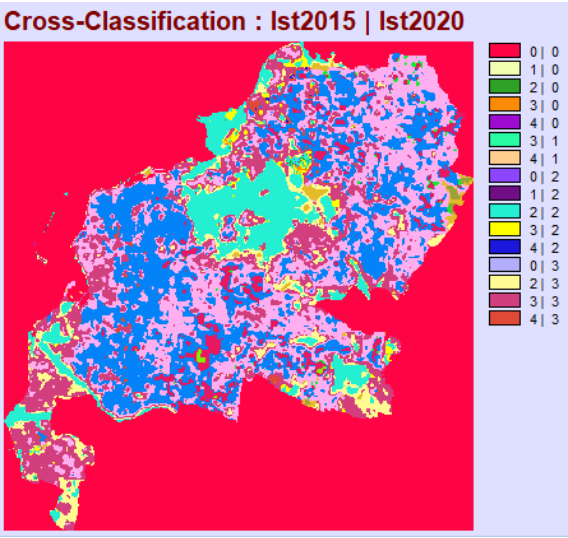


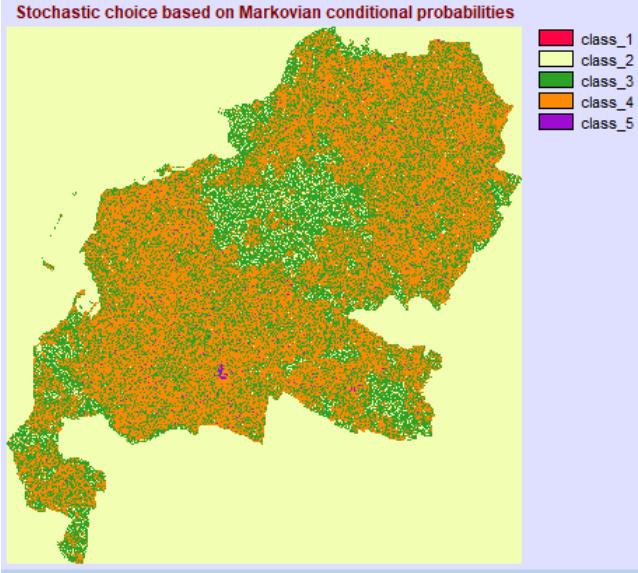


PETA LOKASI PENGUKURAN SUHU UDARA

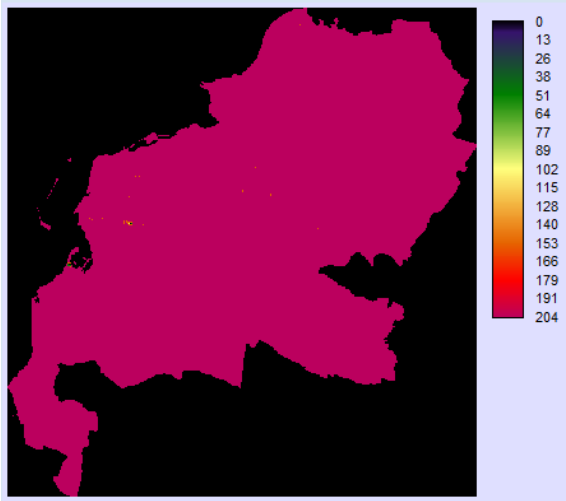




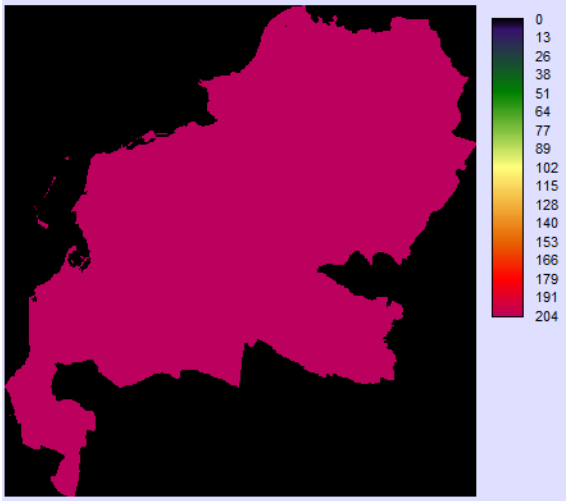




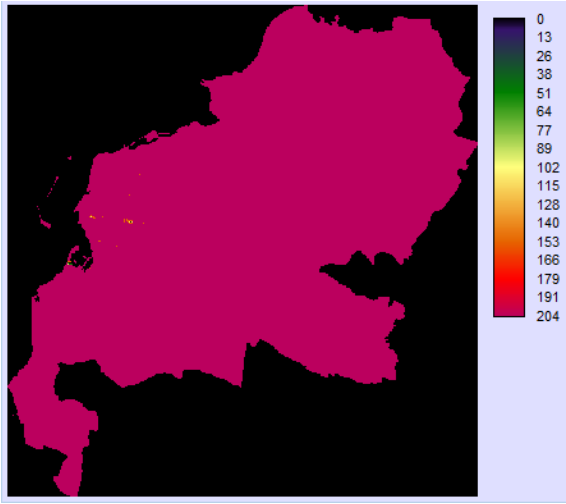
Faktor Pendorong : NDBI tahun 2010



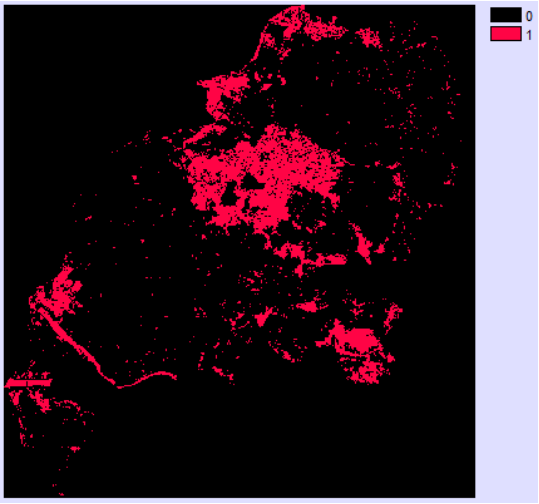
Faktor Pendorong : NDBI tahun 2015



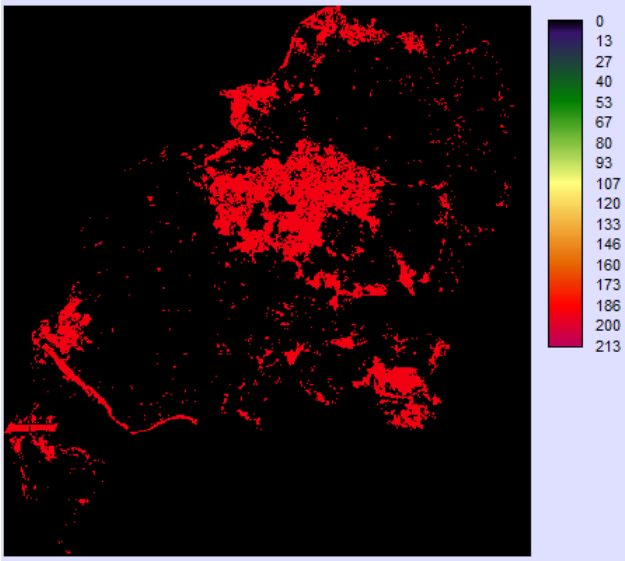
Faktor pendorong : NDVI tahun 2010



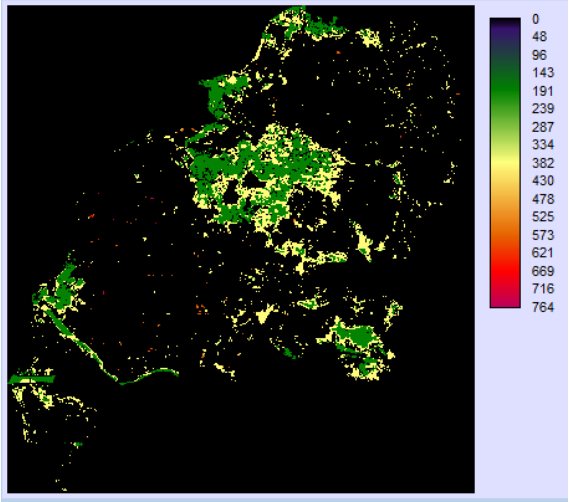
Faktor Pembatas : NDWI tahun 2010



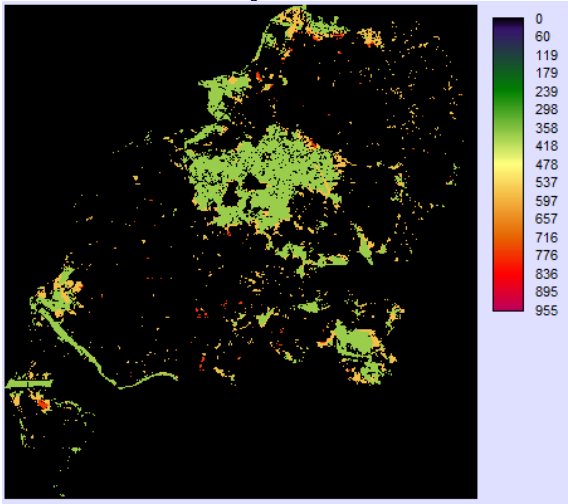
Hasil MCE



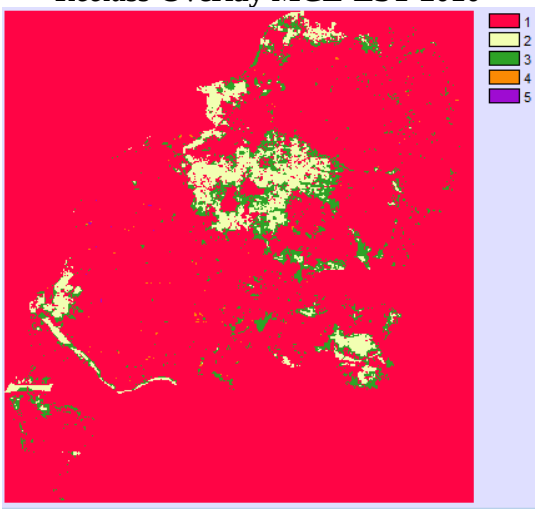
Hasil Overlay MCE- LST 2010



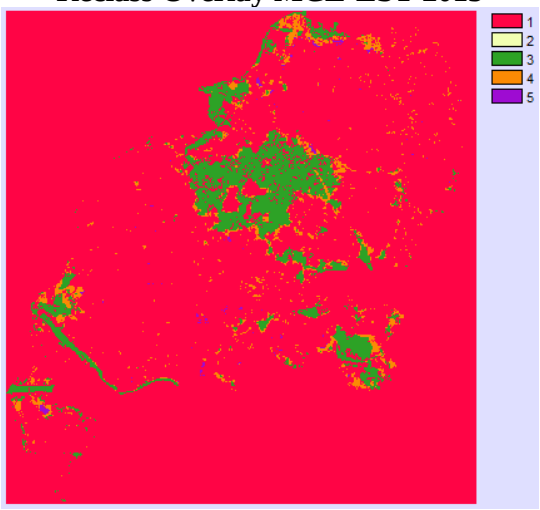
Hasil Overlay MCE-LST 2015

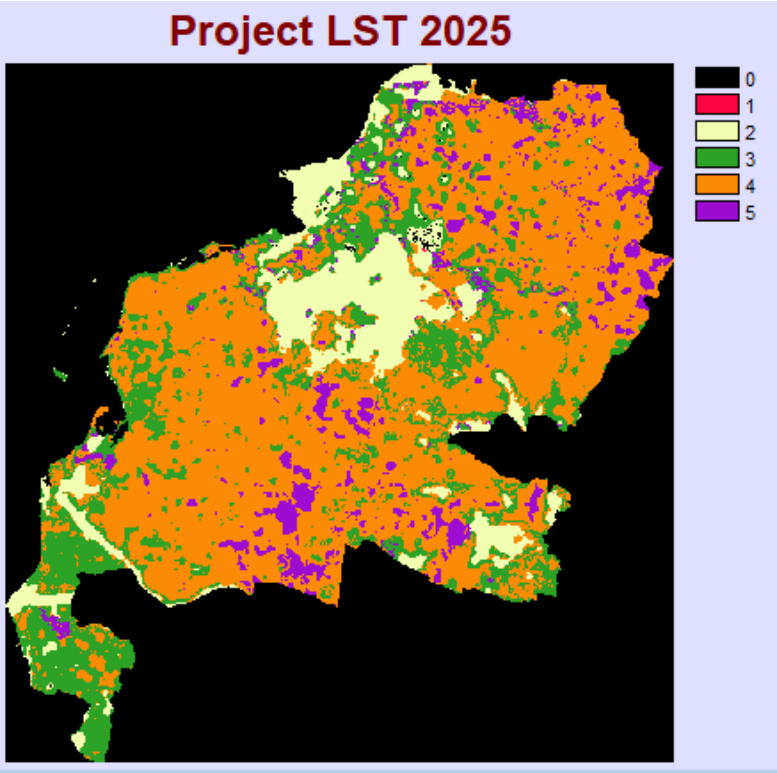


***Reclass* Overlay MCE-LST 2010**



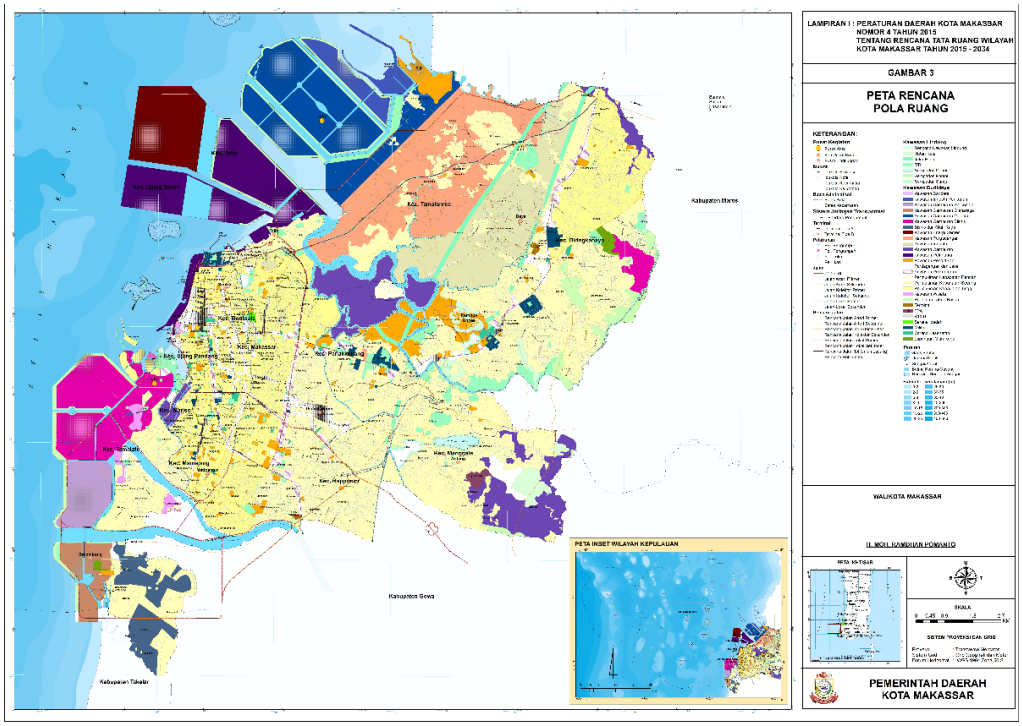
Reclass Overlay MCE-LST 2015



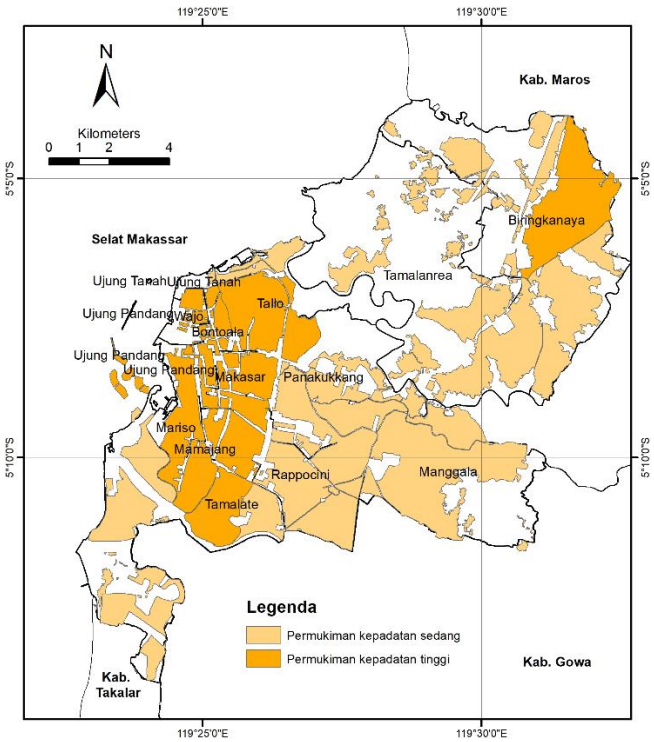


LAMPIRAN 3 : Pengolahan Data SUHII

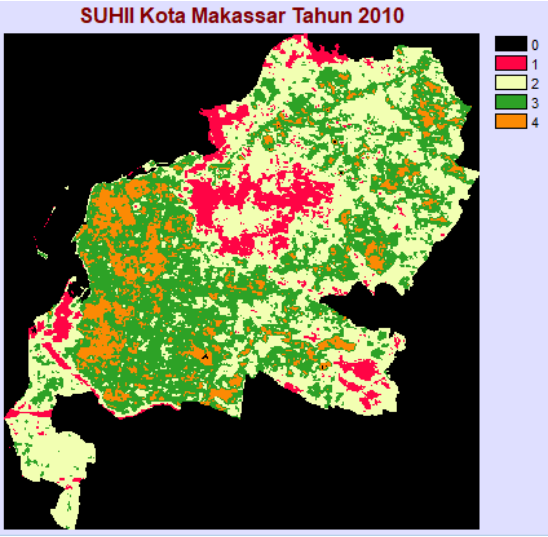
1. Peta Rencana Pola Ruang Kota Makassar tahun 2015-2034 dari Bappeda Kota Makassar

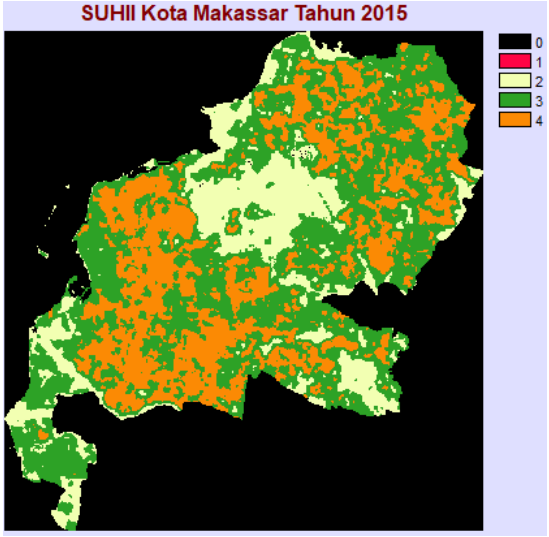


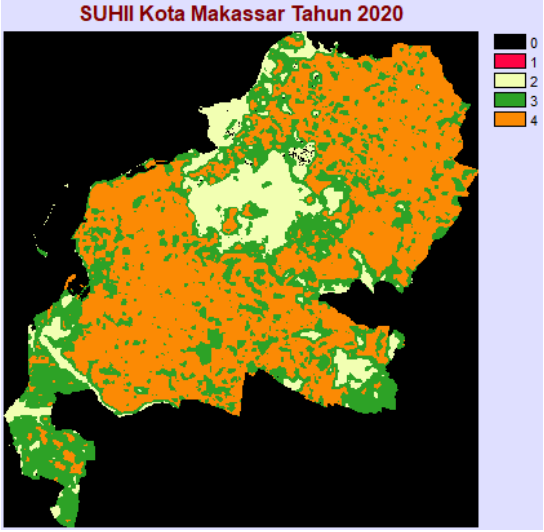
2. Hasil digitasi alokasi rencana permukiman Kota Makassar tahun 2015-2034

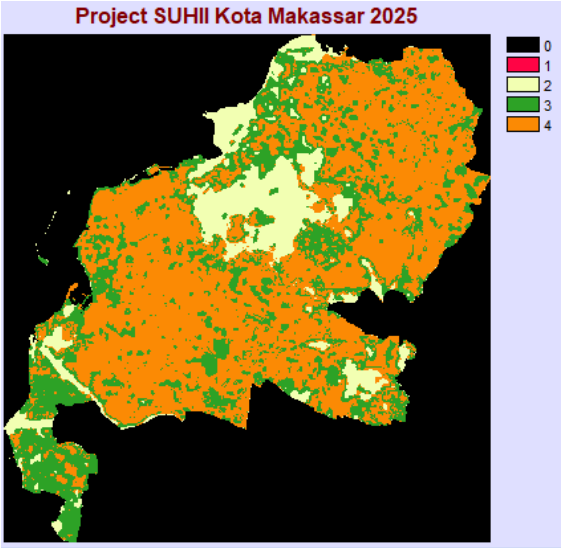


3. **Pengolahan Model Prediksi SUHII tahun 2025**









PETA PREDIKSI SUHII KOTA MAKASSAR TAHUN 2025