**A picture containing clipart

Description automatically generated**

**UNIVERSITAS INDONESIA**

***HYPERPARAMETER TUNING* PADA MODEL *EXTREME GRADIENT BOOSTING* UNTUK KLASIFIKASI CURAH HUJAN: STUDI KASUS KOTA PONTIANAK**

**SKRIPSI**

**AURIWAN YASPER  
1706032004**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
PROGRAM STUDI FISIKA  
DEPOK  
JUNI 2023**

**A picture containing clipart

Description automatically generated**

**UNIVERSITAS INDONESIA**

***HYPERPARAMETER TUNING* PADA MODEL *EXTREME GRADIENT BOOSTING* UNTUK KLASIFIKASI CURAH HUJAN: STUDI KASUS KOTA PONTIANAK**

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains**

**AURIWAN YASPER  
1706032004**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
PROGRAM STUDI FISIKA  
DEPOK  
JUNI 2023**

## HALAMAN PERNYATAAN ORISINILITAS

**Skripsi ini adalah hasil karya saya sendiri,  
dan semua sumber baik yang dikutip maupun dirujuk  
telah saya nyatakan dengan benar**

|  |  |
| --- | --- |
| **Nama** | **: Auriwan Yasper** |
| **NPM** | **: 1706032004** |
| **Tanda Tangan** | **:** |
| **Tanggal** | **: 1 Juni 2023** |

## HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

|  |  |
| --- | --- |
| Nama | : Auriwan Yasper |
| NPM | : 1706032004 |
| Program Studi | : Fisika |
| Judul Skripsi | : *Hyperparameter Tuning* pada *eXtreme Gradient Boosting* untuk klasifikasi curah hujan: Studi kasus Kota Pontianak |

**Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia**

**DEWAN PENGUJI**

|  |  |  |
| --- | --- | --- |
| Pembimbing I | : Dr. Djati Handoko, S.Si., M.Si. | (…………………………) |
| Pembimbing II | : Maulana Putra, S.Si., M.T. | (…………………………) |
| Penguji I | : Drs. Sastra Kusuma Wijaya, Ph.D. | (…………………………) |
| Penguji II | : Dr.rer.nat. Martarizal, | (…………………………) |

|  |  |
| --- | --- |
| Ditetapkan di | : Depok |
| Tanggal | : 1 Juni 2023 |

## KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Sains Jurusan Fisika pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

1. Orang tua penulis, *amak* dan *abak*, yang telah memberikan segenap jiwa dan raganya demi kelancaran hidup saya sedari kecil hingga titik ini, dan pengorbanan yang tak terkira telah diberikan demi berlanjutnya pendidikan saya di perguruan tinggi. Meski penulis seringkali mengabaikan orang tua, kasih sayang mereka tak pernah terputus dan selalu mendoakan penulis hingga detik ini. Tidak lupa, *uda* Ceang, *uni* Mimi, *uda* Ino, *uda* Adi, dan Nana selaku saudara-saudara penulis, dan yang teristimewa *uda* Adi yang telah meluangkan waktunya mempersiapkan penulis menghadapi siding.
2. Dr. Djati Handoko dan Maulana, S.Si., M.T., selaku pembimbing selama berjalannya tugas akhir yang telah memberikan penerangan bagi penulis dalam penyelesaian tugas akhir ini.
3. Drs. Sastra Kusuma Wijaya, Ph.D. dan Dr.rer.nat. Martarizal, selaku dosen penguji sidang skripsi, yang telah memberikan masukan, nasihat dan meluangkan waktunya untuk berdiskusi dalam penyelesaian tugas akhir ini.
4. Filham Pratama Kusuma, S.Si. dan Abdul Aziz, S.Si. selaku kenalan lama yang telah memberikan dukungannya secara moral ketika saya hampir menyerah, dan bersedia membantu saya menyelesaikan masalah yang saya hadapi.
5. Ibu Emiliana Fibriyanti, Ustadz Raziqun, Mas Iqbal, Mbak Nurul, Mas Misran, Ustadz Haidir dan Ustadz Khanzan selaku pembina Pondok Kos Alkahfi yang telah memberikan dukungan moral dan tempat kos yang lebih dari cukup. Terima kasih yang sebesar-besarnya kepada Bu Emi dan keluarga atas dukungannya.
6. Rudi Junas Saputra selaku teman seperjuangan yang telah memberikan dukungan baik moril maupun materil.
7. Teman-teman Pondok Kos Alkahfi (Bang Ridwan senior, Sidiq Almubarak, Bang Andi, Fahem, Salman, Ikhsan, Alim, Guren, Tomi, Romi, Budiman, Khair, Hafizh, Karim, Sayyid, Ihsan, Fatkhur, Faiz, Teddy, Akhyar, Ridwan junior, Taqi, Edi, Izzam, dan Bang Firman) yang selalu mendukung dalam bentuk moril maupun materil.
8. Bang Noval, Bang Amaik, Bang Rahmat, Bang Agum, Bang Ryan, Ridho, Rahman, Popo dan teman-teman IMAMI lainnya, yang tidak bisa penulis sebutkan satu persatu, selaku teman yang selalu memberikan dukungan demi kelancaran tugas akhir penulis.
9. Teman-teman Fisika, FMIPA serta dosen dan staff yang telah membantu saya berkembang di kampus UI, yang tidak bisa saya sebutkan satu persatu.
10. Rekan-rekan bimbingan bapak Dr. Djati Handoko dan Mas Maulana yang senantiasa mendukung dalam melaksanakan penelitian dan penulisan tugas akhir ini.
11. Teman-teman Bright Scholarship Angkatan 3 (Sidiq, Grandy, Surez, Firman, Rafi, Hasbi) yang telah menemani dan membantu penulis berkembang selama 2 tahun lebih di Rumah Pemberdayaan Masyarakat Bright Scholarship.
12. Semua kenalan penulis yang turut meberi dukungan (nisa, ica, nifa, caesar, bang oki zaza dan teman-teman lainnya) yang hadir dalam hidup penulis yang tidak bisa disebutkan satu persatu. Terima kasih atas doa dan dukungannya.
13. Kepada penulis sendiri, yang tetap berjuang keras menghadapi masa sulit demi menyelesaikan perkuliahan di Departemen Fisika, beserta skripsi dengan baik. Terima kasih sudah bertahan.

Depok, 1 Juni 2023

Auriwan Yasper

## HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

|  |  |
| --- | --- |
| Nama | : Auriwan Yasper |
| NPM | : 1706032004 |
| Program Studi | : Fisika |
| Fakultas | : Matematika dan Ilmu Pengetahuan Alam |
| Jenis Karya | : Skripsi |

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusif Royalty-Free Right*)** atas karya ilmiah saya yang berjudul:

*Hyperparameter Tuning* pada *eXtreme Gradient Boosting* untuk klasifikasi curah hujan: Studi kasus Kota Pontianak

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

|  |  |
| --- | --- |
| Dibuat di | : Depok |
| Pada tanggal | : 1 Juni 2023 |

Yang menyatakan

(Auriwan yasper)

## ABSTRAK

|  |  |
| --- | --- |
| Nama | : Auriwan Yasper |
| Program Studi | : Fisika |
| Judul | : *Hyperparameter Tuning* pada *eXtreme Gradient Boosting* untuk klasifikasi curah hujan: Studi kasus Kota Pontianak |
| Pembimbing | : Dr. Djati Handoko, S.Si., M.Si.  Maulana Putra, S.Si., M.T. |

Klasifikas curah hujan sangat membantu masyarakat dan instansi terkait dalam mengambil kebijakan seperti pengololaan sumber daya air, transportasi, pertanian dan pencegahan bencana. Model yang sudah pernah digunakan dalam melakukan klasifikasi curah hujan yaitu *XGBoost*, telah terbukti mampu melakukan klasifikasi dengan efektif, namun masih memerlukan *tuning* pada *hyperparameter*-nya untuk meningkatkan performa model. Penelitian ini bertujuan untuk merancang metode klasifikasi curah hujan dengan model *XGBoost* dan menemukan nilai learning rate terbaik untuk klasifikasi curah hujan. Parameter *max depth*, dan *n estimator* ditetapkan berdasarkan penelitian yang sudah pernah dilakukan. Model ini dibangun berdasarkan data historis curah hujan selama 3 bulan setiap jam, yang telah dikumpulkan oleh peralatan *Automated Weather Observed System* (AWOS) di Stasiun Meteorologi Kota Pontianak. Pencarian *hyperparameter* menggunakan metode *coarse to fine*, yaitu pencarian kasar ke pencarian halus. Pencarian kasar menggunakan *RandomizedSearchCV,* sedangkan pencarian halus dengan *GridSearchCV.* Model dievaluasi dengan metrik *Accuracy*, *precision*, *recall*, dan *F1-score*. Evaluasi menunjukkan bahwa model memilki metrik evaluasi yang baik dengan persentase diatas 80% untuk setiap kasus pembagian data. Nilai learning rate terbaik dengan akurasi tertinggi yang didapatkan pada model dengan 2040 dataset adalah pada kasus klasifikasi biner, yaitu sebesar 0.047 dengan akurasi pada data latih 90.25%.

Kata Kunci:

*hyperparameter*, *XGBoost*, curah hujan, *coarse to fine,* klasifikasi *GridSearchCV*, *RandomizedSearchCV*

## ABSTRACT

|  |  |
| --- | --- |
| Name | : Auriwan Yasper |
| Study Program | : Physic |
| Title | : *Hyperparameters* Tuning In *eXtreme Gradient Boosting* model forrainfall classification: A Case Study in Pontianak City |
| Counselor | : Dr. Djati Handoko, S.Si., M.Si.  Maulana Putra, S.Si., M.T. |

The classification of rainfall is very helpful for the community and related agencies in making policies such as managing water resources, transportation, agriculture, and disaster prevention. The model that has been used to classify rainfall, namely XGBoost, has proven to be able to classify effectively but still requires tuning its hyperparameters to improve model performance. This study aims to design a rainfall classification method using the XGBoost model and find the best learning rate for rainfall classification. The max depth and n estimator parameters are determined based on research that has been done. This model was built based on historical rainfall data for 3 months every hour, which has been collected by the Automated Weather Observed System (AWOS) equipment at the Pontianak City Meteorological Station. The hyperparameter search uses the *coarse-to-fine* method, which is a *coarse-to-fine* search. The coarse search uses RandomizedSearchCV, while the fine search uses GridSearchCV. The model is evaluated with Accuracy, precision, recall, and F1-score metrics. The evaluation shows that the model has good evaluation metrics with percentages above 80% for each case of data sharing. The best learning rate value with the highest accuracy obtained in the model with the 2040 dataset is in the binary classification case, which is equal to 0.047 with an accuracy of 90.25% of the training data.

Key words:

hyperparameter, XGBoost, precipitation, coarse to fine, classification GridSearchCV, RandomizedSearchCV

## DAFTAR ISI

[HALAMAN PERNYATAAN ORISINILITAS ii](#_Toc135029657)

[HALAMAN PENGESAHAN iii](#_Toc135029658)

[KATA PENGANTAR iv](#_Toc135029659)

[HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS vi](#_Toc135029660)

[ABSTRAK vii](#_Toc135029661)

[ABSTRACT viii](#_Toc135029662)

[DAFTAR ISI ix](#_Toc135029663)

[DAFTAR GAMBAR xi](#_Toc135029664)

[DAFTAR TABEL xii](#_Toc135029665)

[DAFTAR LAMPIRAN xiii](#_Toc135029666)

[BAB 1 PENDAHULUAN 1](#_Toc135029667)

[1.1 Latar Belakang 1](#_Toc135029668)

[1.1 Perumusan Masalah 2](#_Toc135029669)

[1.2 Tujuan Penelitian 2](#_Toc135029670)

[1.3 Manfaat Penelitian 3](#_Toc135029671)

[1.4 Batasan Penelitian 3](#_Toc135029672)

[1.5 Sistematika Penulisan 3](#_Toc135029673)

[BAB 2 TINJAUAN PUSTAKA 5](#_Toc135029674)

[3.1 Landasan Teori 5](#_Toc135029675)

[3.2 Penelitian Terkait 12](#_Toc135029676)

[BAB 3 METODE PENELITIAN 35](#_Toc135029677)

[3.1 Alur Penelitian 35](#_Toc135029678)

[3.2 Tahapan Penelitian 36](#_Toc135029679)

[3.3 Tempat Penelitian 38](#_Toc135029680)

[3.4 Jadwal Penelitian 38](#_Toc135029681)

[3.5 Kesimpulan 39](#_Toc135029682)

[BAB 4 HASIL DAN PEMBAHASAN 40](#_Toc135029683)

[4.1 *Data Understanding* 40](#_Toc135029684)

[4.2 *Data Preparation* 45](#_Toc135029685)

[4.3 Implementasi Model *eXtreme Gradient Boosting* 48](#_Toc135029686)

[4.4 Hasil Tuning Hyperparameter 52](#_Toc135029687)

[4.5 Evaluasi Model 55](#_Toc135029688)

[BAB 5 KESIMPULAN DAN SARAN 57](#_Toc135029689)

[5.1 Kesimpulan 57](#_Toc135029690)

[5.2 Saran 57](#_Toc135029691)

[DAFTAR PUSTAKA 58](#_Toc135029692)

## DAFTAR GAMBAR

[Gambar 2. 1 Algoritma Gradient Boosting(Jolly, 2018) 7](#_Toc134949640)

[Gambar 2. 2 Perbedaan grid search dan random search(Deshpande & Kumar, 2018). 11](#_Toc134949641)

[Gambar 2. 3 Teknik Coarse to fine pada random search(Deshpande & Kumar, 2018). 12](#_Toc134949642)

[Gambar 2. 4 Hasil pelatihan dan tes RMSE selama iterasi(Anwar et al., 2021). 13](#_Toc134949643)

[Gambar 2. 5 Ranking masing-masing atribut(Anwar et al., 2021) 15](#_Toc134949644)

[Gambar 2. 6 RR prediksi versus RR sebenarnya(Anwar et al., 2021). 16](#_Toc134949645)

[Gambar 2. 7 Auto Correlation Function plot untuk data training (Agata & Jaya, 2019). 17](#_Toc134949646)

[Gambar 2. 8 Prediksi versus data aktual(Agata & Jaya, 2019) 18](#_Toc134949647)

[Gambar 2. 9 Grafik prediksi vs aktual pada data tes(Agata & Jaya, 2019). 19](#_Toc134949648)

[Gambar 2. 10 Skor kepentingan fitur dataset (Ma et al., 2020). 21](#_Toc134949649)

[Gambar 2. 11 Hasil penyetelan hyperparameter untuk XGBoost dan RF dievaluasi oleh . Proses penyetelan (a) N estimator di XGBoost dan RF, (b) max depth dan minimum child weight di XGBoost, dan (c) maximum depth dan minimum samples split dari Random Forest. (d) Peringkat kepentingan fitur diperoleh dari model XGBoost terlatih. Skor kepentingan dinormalisasi dengan membagi skor maksimum, dan skor maksimum ditetapkan menjadi 100(X. Li et al., 2022). 35](#_Toc134949650)

[Gambar 3. 1 Diagram alur penelitian 37](#_Toc134949651)

[Gambar 4. 1 Ilustrasi penaganan missing value 47](#_Toc134949716)

[Gambar 4. 2 Gambar model *XGBClassifier* secara default 51](#_Toc134949717)

[Gambar 4. 3 Inisialisai rentang pencarian hyperparameter 52](#_Toc134949718)

[Gambar 4. 4 Implementasi kode pada RandomSearchCV 53](#_Toc134949719)

[Gambar 4. 5 Implementasi GridSearchCV 53](#_Toc134949720)

[Gambar 4. 6 Visualisasi pencarian learning rate dengan teknik coarse to fine. a) binary class 60%-40%, b) binary class 70%-30%, c) binary class 80%-20%, d) binary class 90%-10%, e) multiclass 60%-40%, f) multiclass 70%-30%, g) multiclass 80%-20%, h) multiclass 90%-10% 56](#_Toc134949721)

## DAFTAR TABEL

[Table 2. 1 Fitur yang ada pada data cuaca (Anwar et al., 2021) 13](#_Toc134949660)

[Table 2. 2 Pelatihan RMSE dengan n\_rounds (n\_estimators) = 100, n\_fold = 10, eta (learning\_rate) = 0.3, max\_depth = 6 (Anwar et al., 2021). 15](#_Toc134949661)

[Table 2. 3 Perbandingan masing-masing model(Agata & Jaya, 2019). 18](#_Toc134949662)

[Table 2. 4 Dataset yang digunakan dalam penelitian (Ma et al., 2020). 20](#_Toc134949663)

[Table 2. 5 Tabel evaluasi model(Ma et al., 2020). 21](#_Toc134949664)

[Table 2. 6 Kombinasi ruang pencarian dan jumlah komputasi masing-masing metode tuning (Qin et al., 2021). 23](#_Toc134949665)

[Table 2. 7 Parameter hasil tuning yang akan digunakan dalam penelitian(Qin et al., 2021)*.* 23](#_Toc134949666)

[Table 2. 8 Tabel hasil performa model untuk dataset austria(Qin et al., 2021) 24](#_Toc134949667)

[Table 2. 9 Pemilihan tuning hyperparameter (Kavzoglu & Teke, 2022) 27](#_Toc134949668)

[Table 2. 10 hasil tuning hyperparameter dan hasil uji AUC(Hasan et al., 2021) 28](#_Toc134949669)

[Table 2. 11 perbandingan performa masing-masing model untuk dataset kekuatan tekan(Nguyen et al., 2021) 31](#_Toc134949670)

[Table 2. 12 perbandingan performa masing-masing model untuk dataset kekuatan tarik(Nguyen et al., 2021) 32](#_Toc134949671)

[Table 2. 13 hasil tuning hyperparameter optimal(Kaushik & Birok, 2021) 33](#_Toc134949672)

[Table 2. 14 Perbandingan performa(Dalal et al., 2022) 35](#_Toc134949673)

[Tabel 3. 1 Tabel agenda dan jadwal penelitian 40](#_Toc134949820)

[Tabel 4. 1 Tabel dataset yang digunakan dalam penelitian. 43](#_Toc134949782)

[Tabel 4. 2 Statistik Dataset 44](#_Toc134949783)

[Tabel 4. 3 Penjelasan masing-masing variable dan nilai yang hilang 44](#_Toc134949784)

[Tabel 4. 4 Tabel klasifikasi data curah hujan berdasarkan World Meteorogical Organization (WMO) 45](#_Toc134949785)

[Tabel 4. 5 Tabel klasifikasi data secara keseluruhan 46](#_Toc134949786)

[Tabel 4. 6 Keseluruhan dataset 46](#_Toc134949787)

[Tabel 4. 7 *one-hot encoding* fitur kategori pada dataset 48](#_Toc134949788)

[Tabel 4. 8 Tabel pembagian dataset *binary classification* dan *multiclass classification* 49](#_Toc134949789)

[Tabel 4. 9 Tabel distribusi pada data train. 50](#_Toc134949790)

[Tabel 4. 10 penjelasan hyperparameter dan penetapan nilai awalnya 51](#_Toc134949791)

[Tabel 4. 11 penjelasan parameter yang digunakan pada modul RandomSearchCV 52](#_Toc134949792)

[Tabel 4. 14 Hasil tuning hyperparameter 54](#_Toc134949793)

[Tabel 4. 12 Tabel evaluasi binary classification 57](#_Toc134949794)

[Tabel 4. 13 Tabel evaluasi multiclass classification 57](#_Toc134949795)

## DAFTAR LAMPIRAN

[Lampiran 1. Pengkodean klasifikasi *binary* dan *multiclass* 64](#_Toc135024214)

[Lampiran 2. Pengkodean *multiclass classification* 72](#_Toc135024215)

## BAB 1 PENDAHULUAN

### Latar Belakang

Curah hujan yang tinggi sangat mempengaruhi kehidupan manusia di berbagai sektor antara lain pertanian, transportasi, dan juga dapat mengakibatkan bencana alam seperti kekeringan, banjir, dan tanah longsor (Anwar et al., 2021). Peningkatan rata-rata curah hujan secara signifikan menurunkan tingkat kerawanan pangan dalam aktivitas pertanian di suatu daerah, terutama pedesaan (Tankari, 2020). Hal ini menimbulkan dampak positif bagi para petani dalam hal kesuburan lahan pertanian mereka. Berbeda dengan perkotaan yang lebih beresiko terkena bencana banjir. Begitu juga dengan daerah perbukitan, curah hujan sangat berpengaruh terhadap bencana tanah longsor. Menurut data EM-DAT, sejak tahun 1908 hingga 2022, tanah longsor telah menyebabkan kerusakan besar dalam masyarakat dan telah menelan korban jiwa sebesar 67.169 dan kerugian ekonomi lebih dari 11 miliar dolar (Pham et al., 2022). Oleh karena itu, perkiraan cuaca sangat penting untuk menangani bencana yang mungkin akan terjadi.

Perkembangan teknologi dengan sistem yang dapat menganalisa data curah hujan dan melakukan perkiraan cuaca memberikan solusi yang tepat untuk mengatasi berbagai dampak negatif yang dihasilkan dari curah hujan. Algoritma *machine learning* merupakan salah satu teknologi alternatif yang dapat diterapkan. Cukup banyak model *machine learning* yang digunakan dalam mengolah data tersebut untuk mendapatkan prediksi yang akurat akan perkiraan hujan dimasa mendatang, diantaranya *Bayesian methode*, SVM (*Support Vector Machine*), ANN (*Artificial Neural Network*), atau variasinya seperti *Support Vector Regression* (SVR) yang memiliki performa bagus dalam melakukan prediksi curah hujan (Xiang et al., 2018). Pada penelitian ini, model yang dipilih merupakan model yang cukup populer dalam prediksi dengan set data historis yaitu eX*treme Gradient Boosting*.

Untuk memperoleh prediksi dengan performa yang bagus, perlu adanya pengaturan perilaku dari model *machine learning*. Dalam pembelajaran mesin, terdapat sebuah nilai yang dapat mengontrol perilaku dari model yaitu *parameter* dan *hyperparameter*. Selama proses pembelajaran algoritma pembelajaran mempelajari pola-pola pada data dan selalu memperbaharui nilai yang mengontrol perilaku tersebut, nilai inilah yang disebut *parameter*. Setelah model berhasil mempelajari pola data, nilai parameter sudah menjadi bagian dari model tersebut dan tidak dapat diubah diluar pembelajaran. Sebagai contoh, *weight* dan *bias* dalam *neural network* merupakan parameter. Sebaliknya, *hyperparameter* digunakan hanya pada algoritma yang spesifik, dan nilainya diatur sebelum model mempelajari pola data. Nilai dari *hyperparameter* ini akan digunakan untuk memperbaharui *parameter* yang ada pada model selama proses pembelajaran. Sehingga nilai *hyperparameter* sangat mempengaruhi nilai parameter yang terdapat dalam suatu model pembelajaran mesin.

Penyesuaian *hyperparameter* bertujuan untuk mencari kumpulan nilai yang tepat untuk mengoptimalkan algoritma pembelajaran, sambil mengevaluasi nilai parameter yang disesuaikan hingga memperoleh nilai yang optimal. Kumpulan nilai *hyperparameter* tersebut ditujukan untuk mengoptimalkan kinerja model, mengurangi fungsi *loss* dan untuk mendapatkan hasil yang terbaik dengan *error* yang lebih sedikit. Perlu diperhatikan bahwa algoritma pembelajaran mengoptimalkan fungsi *loss* berdasarkan data *input* dan mencari solusi terbaik dalam pengaturan yang diberikan dan disini *hyperparameter* akan mencari kombinasi secara tepat (Navas, 2022).

### Perumusan Masalah

1. Bagaimana cara membuat rancangan metode klasifikasi curah hujan menggunakan model *eXtreme Gradient Boosting*?
2. Bagaimana cara mendapatkan kombinasi *hyperparameter* yang terbaik untuk model *eXtreme Gradient Boosting* dari data curah hujan?

### Tujuan Penelitian

1. Membuat rancangan metode klasifikasi curah hujan menggunakan model *eXtreme Gradient Boosting.*
2. Mendapatkan nilai terbaik dalam penyesuaian kombinasi *hyperparameter* yang tepat untuk memaksimalkan model *Extreme Gradient Boosting*.

### Manfaat Penelitian

Harapannya penelitian ini dapat diterapkan pada:

1. Pertanian: Penentuan waktu tanam dan waktu panen, dan pengeringan padi (padi harus dijemur sebelum diolah jadi beras) sangat bergantung dengan curah hujan
2. Penerbangan: Keselamatan dalam penerbangan juga dipengaruhi oleh besar curah hujan.
3. Aktivitas nelayan: Nelayan yang akan menangkap ikan perlu mengetahui estimasi curah hujan, untuk menghindari gelombang pasang / badai.

### Batasan Penelitian

1. *Hyperparameter* yang akan di-*tuning* adalah *max\_depth, learning\_rate,* dan *n\_estimator* dengan fokus pada mencari nilai *learning\_rate* terbaik*.*
2. Data yang digunakan sebagai target adalah data klasifikasi curah hujan.
3. Data yang digunakan sebagai fitur adalah temperatur, kelembapan, kecepatan angin, dan tekanan udara Kota Pontianak.

### Sistematika Penulisan

Sistematika penulisan naskah penelitian terdiri dari lima bab, diantaranya:

|  |  |
| --- | --- |
| **BAB I** | **PENDAHULUAN**  Membahas latarbelakang, perumusan masalah, tujuan penelitian, manfaat penelitian, dan batasan masalah. |
| **BAB II** | **TINJAUAN PUSTAKA**  Berisi literatur mengenai landasan teori yang digunakan sebagai acuan pada penelitian, dan penelitian terkait yang sudah pernah dilakukan. |
| **BAB III** | **METODE PENELITIAN**  Berisi tentang alur penelitian, tahapan, dan metode apa saja yang digunakan dalam penelitian ini. |
| **BAB IV** | **PEMBAHASAN**  Berisi data-data yang diperoleh dan analisis terhadap hasil penelitian. |
| **BAB V** | **KESIMPULAN DAN SARAN**  Berisi tentang kesimpulan dan saran untuk penelitain selanjutnya. |

## BAB 2 TINJAUAN PUSTAKA

### Landasan Teori

#### *Machine Learning*

*Machine Learning* adalah sekumpulan metode yang dapat mendeteksi pola dalam data dan menggunakan pola tersebut untuk membuat prediksi di masa mendatang. Pembelajaran mesin telah mendapatkan nilai yang sangat besar di berbagai industri, mulai dari keuangan hingga perawatan kesehatan (Jolly, 2018). Istilah *Machine Learning* pertama kali dipopulerkan oleh Arthur Samuel pada tahun 1959. Arthur mengatakan bahwa *machine learning* merupakan cabang ilmu yang mempelajari bagaimana komputer dapat belajar sendiri tanpa harus diprogram secara explisit.

Secara garis besar, *machine learning* dapat dikategorikan menjadi tiga jenis utama:

* *Supervised Learning*: yaitu bentuk pembelajaran mesin di mana data kita dilengkapi dengan sekumpulan label atau variabel target yang berupa angka. Label/kategori ini biasanya milik satu fitur/atribut, yang umumnya dikenal sebagai variabel target. Misalnya, setiap baris data Anda dapat termasuk dalam kategori Sehat atau Tidak Sehat.
* *Unsupervised Learning*: yaitu salah satu bentuk pembelajaran mesin di mana algoritma mencoba mendeteksi/menemukan pola dalam data yang tidak memiliki variabel hasil/target. Dengan kata lain, kita tidak memiliki data yang disertakan dengan label yang sudah ada sebelumnya. Dengan demikian, algoritma biasanya akan menggunakan metrik seperti jarak untuk mengelompokkan data bersama-sama, bergantung pada seberapa jauh mereka satu sama lain.
* *Reinforcement Learning*: *Reinforcement Learning* didasarkan pada teori psikologi, yang dijabarkan setelah serangkaian percobaan yang dilakukan pada hewan. Tahap pertama adalah mendefinisikan tujuan yang ingin dicapai, *Reinforcement Learning* mencoba memaksimalkan imbalan yang diterima untuk pelaksanaan tindakan atau serangkaian tindakan yang memungkinkan kita mencapai tujuan yang telah ditentukan. *Reinforcement Learning* adalah sektor pembelajaran mesin yang sangat menarik, digunakan dalam segala hal mulai dari mobil otonom hingga video game. Ini bertujuan untuk membuat algoritma yang dapat belajar dan beradaptasi dengan perubahan lingkungan.

#### *Gradient Boosting*

Algoritma yang menggunakan teknik *boosting* akan membangun model dari data latih dan membuat model tambahan yang berfungsi untuk memperbaiki kesalahan dari model pertama. Model-model ini terus ditambahkan hingga data latih terprediksi dengan baik atau telah mencapai batas jumlah maksimal model yang ditentukan.

Algoritma *boosting* berfungsi untuk meningkatkan performa atau akurasi prediksi dengan menggabungkan beberapa model sederhana yang dianggap lemah (*weak learners*) menjadi sebuah model yang kuat (*strong ensemble learner*). Asal-usul dari algoritma ini berasal dari algoritma sederhana seperti regresi linier dan decision tree yang dimodifikasi untuk meningkatkan performa. Terdapat dua metode dalam algoritma boosting yaitu Adaptive boosting dan Gradient boosting, yang masing-masing memiliki cara untuk memperbaiki kesalahan pada model sebelumnya. Nama "boosting" merujuk pada fungsinya untuk meningkatkan performa atau akurasi prediksi (Kurikulum developer dicoding, 2023).

Algoritma *Adaptive boosting* dibangun untuk mengatasi masalah klasifikasi, sedangkan *Gradient boosting* dapat digunakan untuk mengatasi masalah regresi dan klasifikasi (Tattar, 2018). Oleh karena itu pada penelitian kali ini, model yang akan digunakan adalah salah satu pengembangan dari *gradient boosting* yaitu *eXtreme Gradient Boosting*.

*Gradient* dalam *Gradient Boosting* mengacu pada perbedaan antara nilai aktual dan prediksi, dan *Boosting* mengacu pada peningkatan, yaitu, memperbaiki error pada iterasi yang berbeda (Ciaburro et al., 2018). Pengklasifikasian pada model *Adaptive Boosting,* bobot ditambahkan ke contoh yang diprediksi oleh model dengan benar. Namun, dalam *Gradient Boosting*, kesalahan residual juga digunakan sebagai label di setiap pohon untuk membuat prediksi di masa mendatang. Konsep ini bisa dilihat pada gambar 2.1:

Diagram

Description automatically generated

Gambar 2. 1 Algoritma Gradient Boosting(Jolly, 2018)

Inilah yang terjadi pada diagram tersebut:

1. *Decision tree* pertama dilatih dengan data yang kita miliki, dan variabel Y sebagai target.
2. Kemudian hitung kesalahan residual untuk *tree* tersebut. Kesalahan residual merupakan selisih antara nilai *actual* dengan nilai prediksi.
3. *Tree* kedua sekarang dilatih, menggunakan residual sebagai target.
4. Proses membangun banyak pohon ini bersifat iteratif, dan berlanjut untuk jumlah *estimator* yang kita tetapkan.
5. Prediksi akhir dibuat dengan menjumlahkan nilai target yang diprediksi oleh *tree* pertama ke hasil penyusutan dan residual untuk semua *tree* lainnya. Penyusutan adalah faktor yang kita gunakan untuk mengontrol laju proses *Gradient boosting* yang kita inginkan.
6. Nilai penyusutan (*learning\_rate*) yang kecil menunjukkan bahwa algoritma akan belajar lebih cepat, dan oleh karena itu, harus dikompensasi dengan jumlah *estimator* dasar yang lebih besar (*decision tree*) untuk mencegah *overfitting*.
7. Nilai penyusutan (*learning\_rate*) yang lebih besar menunjukkan bahwa algoritma akan belajar lebih lambat, sehingga membutuhkan lebih sedikit *tree* untuk mengurangi waktu komputasi (Jolly, 2018).

#### *Extreme Gradient Boosting*

Pada tahun 2014, Dr. Tianqi Chen yang berasal dari universitas washington mengembangkan sebuah model bernama *eXtreme Gradient Boosting* yang menjadikan *gradient boosting* sebagai dasar dalam membangun teorinya (Zhang & Gong, 2020). *Gradient boosting* merupakan algoritma yang dapat meminimalisir fungsi *loss* dan manjadi solusi untuk optimasi berbagai masalah khususnya pada regresi, klasifikasi dan ranking. Pada dasarnya, algoritma XGBoost mengadopsi konsep menyesuaikan parameter pembelajaran secara berulang untuk mengurangi nilai loss function (sebuah mekanisme evaluasi untuk model). XGBoost memanfaatkan model dengan struktur pohon regresi yang lebih teratur untuk memperbaiki kinerja dan mengurangi kompleksitas model agar tidak mengalami *overfitting*. Dengan menggunakan model pohon regresi yang lebih teratur, XGBoost mampu memberikan performa yang lebih baik (Sunata et al., 2020). Hasil prediksi akhir dari *XGBoost* adalah penjumlahan hasil prediksi dari setiap pohon regresi (S. Li & Zhang, 2020). Algoritma berbasis *decision tree* memiliki kinerja yang baik pada data dengan fitur kategorikal dan tidak terlalu berpengaruh terhadap data dengan kelas tidak seimbang (Nyoman et al., 2020). Dalam metode ini, diperlukan sebuah fungsi objektif yang berguna untuk mengevaluasi sejauh mana model yang dihasilkan sesuai dengan data latih. Fungsi objektif ini memiliki dua bagian penting, yaitu nilai yang hilang pada pelatihan dan nilai regularisasi, seperti yang terlihat pada persamaan 2.1.

Dimana L merupakan fungsi *loss*, dan Ω sebagai fungsi regularisasi, dan merupakan parameter dari model. Fungsi pelatihan yang hilang secara umum dapat ditulis seperti pada persamaan 2.2.

Dimana 𝑦𝑖 adalah nilai data *actual* dan merupakan nilai prediksi, sedangkan n merupakan banyak perulangan dalam model.

Diasumsikan data adalah , jadi ada n observasi. Setiap observasi memiliki m fitur dan variabel y. Maka nilai prediksi yang akan diperoleh model adalah , adalah seperti persamaan 2.3:

Disini adalah *regression tree*  dan adalah skor yang disediakan *k-tree* untuk observasi ke-i. Fungsi dipilih untuk meminimumkan nilai fungsi objektif seperti persamaan 2.4:

Dimana *l* adalah *loss function* dan Ω menunjukkan fungsi regularisasi seperti persamaan 2.5:

Dimana T adalah jumlah *leaves*  dan *w* adalah besar bobot *leaf.* Untuk menghindari *overfitting* dan model yang sederhana, pinalti untuk T diatur oleh , dan pinalti untuk *w* diatur oleh Hal yang membedakan dengan *gradient boosting* biasa adalah pinalti yang bernilai unik. Metode iterasi digunakan untuk mengurangi fungsi objektif. Dalam iterasi ke- *t,* maka ditambahkan untuk mengurangi fungsi objektif seperti persamaan 2.6:

Ekspansi Taylor digunakan untuk menyederhanakan persamaan ini. Dari persamaan Taylor kita bisa menurunkan rumus untuk mengurangi *loss* setelah *tree* dibagi dari *node* yang diberikan (Ma et al., 2020), rumusnya bisa dilihat pada persamaan 2.7:

Dimana *l* mewakili subset dari pengamatan yang tersedia di *node* saat ini. adalah hipunan bagian dari observasi yang tersedia di *node* kiri dan merupakan hipunan bagian dari observasi yang tersedia di *node* kanan setelah *split*. Fungsi dan digunakan untuk menemukan *split* terbaik, dan didefenisikan sebagai persamaan 2.8 dan 2.9:

Fungsi objektif akhir hanya bergantung pada gradien orde pertama dan kedua dari fungsi kerugian pada setiap titik data dan parameter regularisasi γ

Selanjutnya *hyperparameter tuning* untuk *XGBoost* adalah sebagai berikut:

1. *max\_depth* dan *min\_child\_weight*: berfungsi untuk mengontrol arsitektur *tree*. *max\_depth* menentukan jumlah *node* maksimum dari *root* hingga *leaf* terjauh (angka defaultnya adalah 6). *min\_child\_weight* adalah *weight* minimum untuk membuat simpul baru *tree*.
2. *learning\_rate*: Ini menentukan jumlah koreksi pada setiap langkah, mengingat setiap putaran *boosting* mengoreksi kesalahan putaran sebelumnya. *learning\_rate* mengambil nilai dari 0 hingga 1, dan nilai defaultnya adalah 0,3.
3. *n\_estimators*: Ini menentukan jumlah pohon dalam *ensembel*. Nilai defaultnya adalah 100. Perhatikan bahwa jika menggunakan *vanilla XGBoost* dari *scikit-learn*, kita akan menggunakan *num\_boost\_rounds* daripada *n\_estimators*.

#### *Hyperparameter Tuning*

Bayangkan sebuah *sound system* yang memiliki sistem *speaker* dan *mixer* berkualitas tinggi. Anda pasti pernah melihat serangkaian tombol di konsol yang secara independen mengontrol parameter kualitas suara tertentu. *Bass*, *treble*, dan kenyaringan adalah beberapa kontrol yang harus diatur dengan benar untuk pengaturan suaranya agar enak didengar. Demikian pula, model *machine learning* yang mempunyai pengaturan atau penyesuaian berbagai parameter kontrol. Parameter ini disebut *hyperparameter*, dan proses pengontrolan berbagai parameter pada suatu nilai digunakan untuk mendapatkan kinerja terbaik model dengan waktu pelatihan/eksekusi serta akurasi dan generalisasi model. Mirip dengan contoh *equalizer* suara, beberapa *hyperparameter* perlu disetel bersama untuk performa optimal. Ada dua strategi yang biasanya digunakan saat memilih kombinasi hyperparameter:

1. ***Grid Search***: *Hyperparameter* diplot pada matriks dan kombinasi yang mendapatkan kinerja terbaik dipilih untuk model yang digunakan dalam skenario sebernarnya. *Grid Search* merupakan metode untuk penyesuaian hyperparameter dengan cara "*brute force*". Dalam metode ini, kita membuat kisi dari kemungkinan nilai *hyperparameter* yang bersifat diskrit dan model disesuaikan dengan setiap kombinasi nilai *hyperparameter* tersebut. Performa model untuk setiap kombinasi direkam dan kemudian dipilih kombinasi yang menghasilkan performa terbaik. *Grid Search* adalah algoritma *powerfull* yang dapat menemukan kombinasi terbaik dari hyperparameter secara tepat, akan tetapi metode ini memiliki kelemahan dalam kecepatannya. Menyesuaikan model dengan setiap kombinasi membutuhkan kapasitas komputasi yang tinggi dan waktu yang signifikan, sehingga proses akan sangat lama dan tidak efisien terutama jika sumber daya komputasi terbatas (Navas, 2022).
2. ***Random Search***: Dalam kasus *random search*, nilai *hyperparameter* dipilih secara acak. Dalam hal ini, dengan jumlah iterasi yang sama dengan *grid search*, ada peluang yang lebih baik untuk mencapai nilai optimal untuk *hyperparameter*. Perbedaan antara *grid search* dan *random search* dapat dilihat pada gambar 2.2.

Chart, diagram, scatter chart

Description automatically generated

Gambar 2. 2 Perbedaan grid search dan random search(Deshpande & Kumar, 2018).

Variasi dari teknik *random search* dapat digunakan untuk mengurangi jumlah iterasi melalui ruang pencarian. Teknik ini secara luas dikategorikan sebagai *Coarse to fine search*. Dalam hal ini, pencarian acak dijalankan untuk beberapa iterasi dan setelah wilayah dengan kombinasi pengoptimalan yang lebih tinggi diidentifikasi, ruang pencarian dibatasi pada zona nilai *hyperparameter* yang lebih kecil. Dengan teknik ini, pencarian dibatasi pada suatu wilayah. Teknik *Coarse to fine* dapat divisualisasikan dalam gambar 2.3.

Chart, scatter chart

Description automatically generated

Gambar 2. 3 Teknik Coarse to fine pada random search(Deshpande & Kumar, 2018).

Selama iterasi pencarian awal, seluruh ruang dicari. Saat nilai hyperparameter optimal ditemukan, maka ruang pencarian dibatasi ke zona yang lebih halus. Dengan ini, *hyperparameter* disetel dengan halus dengan jumlah iterasi yang relatif lebih kecil.

### Penelitian Terkait

#### *Rainfall Prediction Using Extreme Gradient Boosting.*

Penelitian mengenai prediksi curah hujan menggunakan eX*treme Gradient Boosting* pernah dilakukan oleh M T Anwar et al. tahun 2021. Data curah hujan yang digunakan berasal dari BMKG Tanjung Mas, Kota Semarang. Terdapat 11 fitur dalam dataset, akan tetapi yang digunakan hanya 8 fitur saja seperti disajikan dalam tabel 2.1. Dataset *training* terdiri dari data cuaca harian dari tahun 2013 hingga 2019, sedangkan data *testing* adalah data cuaca harian tahun 2020. Pada fase *training*, entri dengan *missing value* dihilangkan.

Table 2. 1 Fitur yang ada pada data cuaca (Anwar et al., 2021)

|  |  |  |
| --- | --- | --- |
| **Simbol** | **Tipe Data** | **Keterangan** |
| Tn | *Numeric* | Suhu minimum |
| Tx | *Numeric* | Suhu maksimum |
| Tavg | *Numeric* | Rata-rata suhu |
| RH\_avg | *Numeric* | Rata-rata kelembapan (%) |
| Ss | *Numeric* | *Sun exposure time (hours)* |
| ff\_x | *Numeric* | Kecepatan angin maksimum (m/s) |
| ff\_avg | *Numeric* | Rata-rata kecepatan angin (m/s) |
| RR | *Numeric* | Curah hujan (mm) |

Setelah dilakukan pelatihan model dalam 100 putaran, diperoleh nilai RMSE untuk pelatihan lebih baik daripada nilai saat training. Hal ini ditunjukkan pada gambar 2.4. Hal ini menunjukkan kecenderungan *overvitting* XGBoost seiring bertambahnya iterasi.

Chart

Description automatically generated

Gambar 2. 4 Hasil pelatihan dan tes RMSE selama iterasi(Anwar et al., 2021).

Table 2.2 menunjukkan bahwa nilai kesalahan pelatihan, *10-fold* *cross validation* menghasilkan kesalahan yang lebih rendah daripada menggunakan dataset *full training* yang menunjukkan indikasi *overfitting.* Ketika data non-hujan dikecualikan kesalahannya meningkat. Kesalahan yang lebih tinggi ini mungkin disebabkan oleh hilangnya informasi tentang karakteristik hari tidak hujan. Sehingga mengurangi kemampuan model untuk memprediksi secara akurat jumlah curah hujan.

Table 2. 2 Pelatihan RMSE dengan n\_rounds (n\_estimators) = 100, n\_fold = 10, eta (learning\_rate) = 0.3, max\_depth = 6 (Anwar et al., 2021).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Round** | ***RMSE* pelatiahan** | | | |
| **Dengan data *non-rainy*** | | **Tampa data *non-rainy*** | |
| **10-fold CV** | ***Complete Training*** | **10-fold CV** | ***Complete Training*** |
| 1 | 13.22+0.25 | 13.30 | 22.36+0.39 | 20.46 |
| 50 | 4.41+0.14 | 4.75 | 9.62+0.27 | 12.93 |
| 100 | 2.46+0.11 | 2.75 | 6.92+0.25 | 10.85 |

*XGBoost* mampu memeringkat atribut penting yang berkontribusi pada model. Pemeringkatan atribut ditunjukkan pada Gambar 2.5 Ini menunjukkan bahwa model prediksi curah hujan sangat dipengaruhi oleh kelembaban relatif rata-rata (RH\_avg) dan suhu minimum (Tn). Hasil ini sejalan dengan penelitian sebelumnya yang menggunakan model C4.5 untuk memprediksi apakah suatu hari hujan atau tidak (Anwar et al., 2021).

Chart

Description automatically generated

Gambar 2. 5 Ranking masing-masing atribut(Anwar et al., 2021)

Saat diuji dengan data cuaca pada tahun 2020, model tersebut memberikan MAE 8,8. Gambar 2.6 menunjukkan *scatter plot* dari prediksi RR terhadap RR aktual pada dataset uji. Ini menunjukkan bahwa banyak data terkonsentrasi pada nilai mendekati 0 yang sangat menantang untuk model. Garis biru adalah garis tren linier dengan korelasi R sebesar 0,555. Hasil ini lebih rendah dari penelitian terbaru oleh Lee et al. tahun 2020 yang menggunakan *Nonlinear Autoregressive* Neural Network dan memiliki R = 0,9. Saran penelitian masa depan untuk mengeksplorasi pengaturan parameter (*tuning*) *XGBoost* untuk meningkatkan kemampuan prediksinya. Penelitian lebih lanjut juga dapat menggabungkan data berbasis Bumi dengan data penginderaan jauh untuk membuat model yang lebih akurat.

Chart, scatter chart

Description automatically generated

Gambar 2. 6 RR prediksi versus RR sebenarnya(Anwar et al., 2021).

Pada penelitian ini disimpulkan bahwa model yang dibangun berdasarkan data cuaca historis selama 7 tahun yang dikumpulkan oleh stasiun cuaca menghasilkan prediksi yang akurat untuk estimasi curah hujan harian dengan RMSE 2.7 mm dan MAE pengujian 8.8 mm. Hasil penelitian juga menunnjukkan bahwa faktor yang paling mempengaruhi adalah kelembapan rata-rata dan suhu minimum. Penelitian selanjutnya sangat diharapkan adanya *tuning hyperparameter* *XGBoost* untuk meningkatkan akurasi model, dan tambahan data pengindraan jarak jauh.

#### Studi komparasi model *extreme gradient boosting, SARIMA, exponential smoothing,* dan *Neural Network* untuk peramalan data curah hujan

Penelitian selanjutnya dilakukan oleh Agata et al. pada tahun 2019 melakukan penelitian mengenai perbandingan antara beberapa algoritma *machine learning* yaitu *XGBoost*, SARIMA (*Seasonal Autoregressive Integrated Moving Average*), *exponential smoothing*, dan ANN (*Artificial Neural Network*)*.* Empat model ini dilakukan komparasi untuk mentukan model terbaik untuk melakukan prediksi curah hujan di kota Bandung dari tahun 2018-2019.

Untuk memahami data, langkah awal yang dilakukan peneliti adalah membuat pola sementara untuk data hujan dengan menggunakan *Auto Correlation Function* (ACF). Hasilnya bisa dilihat pada gambar 2.7.

Chart

Description automatically generated

Gambar 2. 7 Auto Correlation Function plot untuk data training (Agata & Jaya, 2019).

Gambar diatas menunnjukkan *Autocorrelation function* dengan pola musim yang jelas dengan *time lag* 12 (s = 12 bulan). Ini menunnjukan kota Bandung memiliki pola musim berulang setiap tahunnya. Metode evaluasi yang digunakan adalah MAD (*Mean Absolute Deviance*), RMSED (*Root Mean Squared Error Deviance*)*,* dan MAPE (*Mean Absolute Percentage Deviance*). Perbandingan evaluasi masing masing model ditunjukkan dalam tabel 2.3.

Table 2. 3 Perbandingan masing-masing model(Agata & Jaya, 2019).

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MAD** | **RMSED** | **MAPE** |
| SARIMA | 116.9017 | 162.6560 | 65.7756 |
| *exponential smoothing* | 117.1898 | 158.5143 | 64.8473 |
| *Artificial Neural Network* | 117.1898 | 158.5143 | 64.8473 |
| *XGBoost* | 112.5225 | 152.8657 | 63.7136 |

Berdasarkan tabel diatas kita ketahui bahwa model dengan nilai MAD, RMSED, dan MAPE terendah adalah *XGBoost.* Selanjutnya tes dilakukan dengan data hujan tahun 2018-2019 dan diperoleh grafik prediksi vs aktual pada gambar 2.8.

A picture containing text, linedrawing

Description automatically generated

Gambar 2. 8 Prediksi versus data aktual(Agata & Jaya, 2019)

Secara jelasnya bisa dilihat pada gambar 2.9.

Chart

Description automatically generated with medium confidence

Gambar 2. 9 Grafik prediksi vs aktual pada data tes(Agata & Jaya, 2019).

#### *Very Short-Term Renewable Energy Power Prediction*

Selanjutnya penelitian yang dilakukan oleh Zhenchuan ma et al. tahun 2020 yang meneliti *XGBoost* sebagai model untuk prediksi kekuatan angin. Selain prediksi yang akurat *XGBoost* juga memiliki kelebihan dalam mengatur *thread* CPUs dalam melaksanakan tugas secara paralel (Ma et al., 2020). Metode *tuning hyperparameter* yang digunakan pada penelitian ini adalah TPE (*Tree-Structured Parzen Estimator*).

Turbin Angin Sotavento merupakan sebuah turbin angin eksperimental, terletak di Galicia, Spanyol dengan koordinat 43.354377oN dan 7.881213oW. Galicia terletak di barat laut Spanyol, dekat samudra Atlantik dan memiliki iklim maritim dan Atlantik. Wilayah ini kaya akan sumber daya angin, khususnya, selalu ada angin kencang di bulan Januari, Februari, dan Maret. Angin yang bertiup datang dari barat daya dan tiba setelah melewati banyak air. Ada 24 turbin angin dengan 9 model yang berbeda di ladang turbin angin. Total daya nominal adalah 17,56MW.

Data kecepatan angin dan arah angin dihasilkan pada menara anemometri. Tenaga angin yang dihasilkan diukur melalui sistem SCADA. Semua data yang digunakan dalam penelitian ini dirata-ratakan dan disimpan dengan interval 10 menit, mulai dari '1 Januari 2015' hingga '1 Januari 2018'. Catatan juga berisi nilai *missing value*, dan jumlah total catatan yang ditunjukkan pada Tabel 2.4 belum memperhitungkan *missing value*. Kumpulan data pelatihan mencakup 52069 data dan digunakan untuk melatih model *XGBoost*. Kumpulan data validasi berisi 12925 data dan digunakan untuk mengevaluasi kinerja model sehingga *hyperparameter* terbaik dapat dipilih. Kumpulan data uji berisi 12230 data dan digunakan untuk menguji kinerja peramalan model.

Table 2. 4 Dataset yang digunakan dalam penelitian (Ma et al., 2020).

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Time Stamp awal** | **Time Stamp akhir** | **Jumlah data** |
| Data Latih | 2016-01-01 00:10 | 2017-01-01 00:00 | 52.069 |
| Data Validasi | 2017-01-01 00:10 | 2017-04-01 00:00 | 12.925 |
| Data Uji | 2017-04-01 00:10 | 2017-07-01 00:00 | 12.230 |

Saat XGBoost melatih model, XGBoost dapat menghitung rata-rata pengurangan loss function setiap fitur pada pemisahan node. Dengan ini, model dapat menghitung kontribusi setiap fitur pada pelatihan model. Semakin tinggi nilainya, semakin penting fitur ini(Ma et al., 2020). Oleh karena itu, setelah model XGBoost dilatih, skor kepentingan setiap fitur dapat langsung diperoleh. Seperti disebutkan sebelumnya, jumlah total fitur kandidat adalah 133. Pada penelitian ini menggunakan fitur ini sebagai input ke algoritma XGBoost untuk melatih model dan hasilnya terdapat skor kepentingan untuk 40 fitur teratas, bisa dilihat pada Gambar 2.10.

Chart, histogram

Description automatically generated

Gambar 2. 10 Skor kepentingan fitur dataset (Ma et al., 2020).

Metode evaluasi data pada penelitian ini adalah dengan menggunakan NRMSE (*Normalized Root Mean Square Error*), NMAPE (*Normalized Mean Absolute Percentage Error*) dan NPE (*Normalized Percentage Error*). Sedangkan metode prediksinya menggunakan tiga komparasi model yaitu *Persistence, Support Vector Regression* (SVR), dan *XGBoost.* Hasil evaluasi untuk masing-masing model dapat dilihat pada tabel 2.5.

Table 2. 5 Tabel evaluasi model(Ma et al., 2020).

|  |  |  |  |
| --- | --- | --- | --- |
| **Kriteria evaluasi** | **Persistensi** | **SVR** | **XGBoost** |
| RMSE | 635.8632 | 522.7257 | 464.2721 |
| MAPE | 322.9744 | 326.2751 | 250.6030 |
| NRMSE | 3.6211 | 2.9768 | 2.6439 |
| NMAPE | 1.8393 | 1.8581 | 1.4271 |

Beradasarkan data tabel diatas dapat kita lihat bahwa model *XGBoost* memiliki akurasi prediksi yang tinggi dengan nilai NRMSE dan MAPE terendah.

***3.2.4* Optimalisasi *XGBoost* oleh *Adaptive Particle Swarm Optimization* untuk Penilaian Kredit**

Permasalahan *credit scoring* merupakan masalah yang cukup menantang. Perkembangan teknologi seperti *machine learning* telah menunjukkan performa yang memuaskan untuk mengatasi masalah kehidupan sehari-hari. Karena itu keuntungan yang disajikan dari kombinasi fitur dan pemilihan fitur, seperti *decision tree* dapat menyesuaikan data kredit yang memiliki dimensi tinggi dan kompleks. Salah satu model yang mengambil gagasan *decision tree* adalah *eXtreme Gradient Boosting* yang memperkuat *tree* untuk mengatasi kekurangan dengan mengintegrasikan model *tree* lainnya. Struktur model ditentukan oleh *hyperparameter*, yang ditujukan pada permasalahan umum yaitu masalah *manual tuning* yang menghabiskan waktu yang cukup lama, karena itu bebarapa metode pengoptimalan digunakan untuk melakukan *tuning*.

*Particle swarm optimization* (PSO) menggambarkan keadaan partikel dan hukum geraknya sebagai bilangan real kontinu, maka *hyperparameter* yang berlaku untuk *eXtreme Gradient Boosting* dapat ditemukan nilai optimalnya dalam ruang pencarian kontinu. Namun, *Particle swarm optimization* klasik cenderung jatuh ke optima lokal. Untuk mengatasi masalah ini, Chao Qin et al. mengusulkan model penilaian kredit *eXtreme Gradient Boosting* yang didasarkan pada *Particle swarm optimization* adaptif. Swarm split didasarkan pada ide pengelompokan dan dua jenis strategi pembelajaran akan digunakan untuk memandu partikel untuk meningkatkan keragaman subswarm, untuk mencegah algoritma jatuh ke optimal lokal.

Dalam penelitian ini, algoritman pembelajaran mesin yang akan digunakan adalah algoritma tradisional dan pengklasifikasi pembelajaran *ensemble* populer, serta empat metode pengoptimalan *hyperparameter* (pencarian grid (*Grid Search*), pencarian acak (*Random Search*), TPE (T*ree-structured Parzen Estimator*), dan PSO (*Particle swarm optimization*)) dan melakukan pembandingan performa masing-masing metode. Evaluasi model dalam penelitian ini dilakukan dengan empat set data kredit dan tujuh set data tolok ukur KEEL pada lima nilai evaluasi yang populer yaitu *accuracy*, *error rate* (erortipe I and eror tipe II), *Brier score*, and F1 *score*. Hasil menunjukkan bahwa model yang diusulkan yaitu *Adaptif Particle Swarm Sptimization* (APSO) mengungguli model lain secara rata-rata. Dalam tabel 2.6 disajikan berapa saja *hyperparameter* yang akan di-*tuning* dan hasil *tuning hyperparameter-*nya disajikan dalam table 2.7 (Qin et al., 2021).

Table 2. 6 Kombinasi ruang pencarian dan jumlah komputasi masing-masing metode tuning (Qin et al., 2021).

|  |  |  |  |
| --- | --- | --- | --- |
| **Hyper parameter** | ***APSO, PSO, TPE, RS*** | ***Grid Search*** | ***Default value*** |
| *Learning\_rate* | 0.1 | 0.1 | 0.1 |
| *Number\_of\_boosts* | 60 | 60 | 60 |
| *Maximum\_tree\_depth* | (1, 12) | 1, 2, 3 | 0.2 ∗ *number of features* |
| *Subsample\_ratio* | (0.9, 1) | 0.9, 0.95, 1 | 0.9 |
| *Column\_subsample\_ratio* | (0.9, 1) | 0.9, 0.95, 1 | 0,9 |
| *Minimum\_child\_weight* | (0, 4) | 0, 1, 2, 3, 4 | 2 |
| *Maximum\_delta\_step* | (0, 1) | 0.4, 0.6, 0.8, 1 | 1 |
| *Gamma* | (0, 0.01) | 0, 0.01 | 1 |
| *Number of computations* | 200 | 2160 | — |

Table 2. 7 Parameter hasil tuning yang akan digunakan dalam penelitian(Qin et al., 2021)*.*

|  |  |
| --- | --- |
| **Algoritma** | **Parameter** |
| *Decision Tree* | *Minimum samples leaf = 6*  *Maximum depth = 8*  *Minimum samples split = 2* |
| *Linear Regression* | Tidak ada pramater khusus |
| *Neural Network* | *Epoch = 1000*  *Learning rate = 0.01* |
| *Support Vector Machine* | *Kernel: RBF*  *C = 32*  *Gamma = 0.1* |
| *Adaptive Boosting*, *Bagging-Decision-Tree*, *Random Forest* | *N estimator = 100*  Siada parameter lainnya sama dengan *Decision Tree* |
| *Adaptive Boosting*-*Neural Network*, *Bagging-Neural Network* | *N estimator = 100*  Sisanya sama sama dengan *Neural Network* |
| *Gradient Boosting Decision Tree* | *N estimators = 100*  *Subsample = 0.9*  *Learning rate = 0.1*  *Min samples leaf = 1*  *Max depth = 7* |

Pengukuran performa dari model untuk salah satu dataset bisa kita lihat pada tabel dibawah

Table 2. 8 Tabel hasil performa model untuk dataset austria(Qin et al., 2021)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Skor F1 | Tipe error II (%) | Tipe error I (%) | ACC (%) | Skor brier |
| *Linear Regression* | 0.7865 | 16.88 | **8.67∗** | 86.77 | 0.1019 |
| *Decision Tree* | 0.8543 | 13.95 | 17.41 | 84.51 | 0.1359 |
| *Neural Network* | 0.8612 | 16.87 | 12.06 | 85.27 | 0.1111 |
| AdaBoost | 0.8644 | 11.93 | 17.40 | 85.64 | 0.1034 |
| SVM | 0.3961 | 13.74 | 15.35 | 85.54 | 0.1012 |
| Bagging-DT | 0.4926 | 13.78 | 13.33 | 86.42 | 0.0987 |
| AdaBoost-NN | 84.59 | 16.36 | 14.22 | 0.8615 | 0.1174 |
| XGBoost | **87.58** | 12.97 | **11.79** | **0.0908** | **0.8756** |
| GBDT | 86.14 | 13.43 | 14.19 | 0.0991 | 0.6026 |
| *Random Forest* | 87.41 | 13.26 | 12.05 | 0.0971 | 0.7571 |
| Bagging-NN | 85.62 | 11.83 | 16.42 | 0.1062 | 0.8683 |

#### *Optimalisasi hiperparameter lanjutan untuk meningkatkan prediksi spasial dari tanah longsor dangkal menggunakan XGBoost*

Algoritma pembelajaran mesin secara progresif telah menjadi bagian dari praktik pemetaan kerentanan tanah longsor karena ketangguhannya dalam menangani mekanisme tanah longsor yang rumit dan non-linier. Namun, struktur internal algoritma semacam itu berisi sekumpulan konfigurasi *hyperparameter* yang pengaturannya benar sangat penting untuk mendapatkan kinerja tertinggi yang dapat dicapai. Penelitian ini menyelidiki keefektifan dan kekokohan algoritma pengoptimalan lanjutan, termasuk *Genetic Algorithm*, *Random Search*, *Bayesian Optimization with Tree-structure Parzen Estimators*, *Bayesian Optimization* *with* *Gausian Process*, dan metode *Hyperband*, untuk mengoptimalkan *hyperparameter* algoritma *eXtreme Gradient Boosting (XGBoost)* dalam prediksi spasial tanah longsor. Terdapat 12 faktor penyebab yang dianggap menghasilkan *landslide susceptibility maps* (LSM) untuk provinsi Trabzon Turki, di mana tanah longsor dangkal translasi terjadi di mana-mana. Metrik akurasi yang akan digunakan untuk mengukur keefektifan strategi pengoptimalan pada algoritme XGBoost adalah *overall accuracy* (OA, *F1-score*), *precision*, A*rea Under the receiver operating characteristic Curve* (AUC), *recall*, dan uji signifikansi statistik.

Dibandingkan dengan model *XGBoost* dengan pengaturan default, model yang dioptimalkan memberikan peningkatan signifikan hingga 13% dalam hal akurasi keseluruhan, yang juga dipastikan dengan uji McNemar. Analisis AUC menunjukkan bahwa model memiliki kinerja yang serupa secara statistik, metode GA (0,942) dan *Hyperband* (0,922) memiliki kemampuan prediksi tertinggi, diikuti oleh *Bayesian Optimization* *with* *Gausian Process* (0,920), *Bayesian Optimization with Tree-structure Parzen Estimators* (0,899), dan *Random Search* (0,894). Analisis efisiensi biaya komputasi menunjukkan bahwa pendekatan *Hyperband* (40,3 detik) jauh lebih cepat (sekitar 13 kali) daripada GA dalam *tuning hyperparameter*, dan dengan demikian algoritma pengoptimalan terbaik untuk masalah ini adalah *hyperband* (Kavzoglu & Teke, 2022). Pemilihan *hyperparameter* bisa dilihat pada tabel 2.9.

Table 2. 9 Pemilihan tuning hyperparameter (Kavzoglu & Teke, 2022)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Hyper parameter** | **Raung pencarion** | | **Nilai Optimal** | | | | |
| **Batas atas** | **Batas bawah** | **BO-GP** | **RS** | **Hyperband** | **GA** | **BO-TPE** |
| Min child weight | 20 | 1 | 17 | 7 | 3 | 16 | 5 |
| Learning rate | 1.0 | 0.1 | 0.3 | 0.7 | 0.5 | 0.1 | 0.6 |
| Maximum depth | 20 | 1 | 3 | 11 | 4 | 1 | 2 |
| Gamma | 0.2 | 0.0 | 0.0 | 0.0 | 0.2 | 0.1 | 0.1 |
| Colsample by tree | 0.7 | 0.5 | 0.5 | 0.5 | 0.7 | 0.7 | 0.6 |
| N estimator | 500 | 100 | 100 | 390 | 17 | 150 | 500 |
| subsample | 1.0 | 0.8 | 0.9 | 1.0 | 1.0 | 0.9 | 0.9 |

#### Mengkaitkan Klasifikasi Serapan Vaksin Campak dan Faktor Pendukungnya Menggunakan Model Ensemble Machine Learning

Campak adalah salah satu masalah kesehatan masyarakat yang signifikan yang bertanggung jawab atas tingginya angka kematian di seluruh dunia, terutama di negara-negara berkembang. Dengan menggunakan data survei demografi dan kesehatan yang representatif secara nasional, penggunaan vaksin campak telah diklasifikasikan, dan faktor-faktor yang mendasarinya diidentifikasi melalui pendekatan *Ensemble Machine Learning* (ML) (Hasan et al., 2021).

Pertama, nilai yang hilang diperhitungkan dengan menggunakan berbagai pendekatan, dan kemudian beberapa teknik pemilihan fitur telah diterapkan untuk mengidentifikasi atribut penting untuk memprediksi vaksinasi campak. Teknik pengoptimalan *hyperparameter* yaitu *grid search* telah diterapkan untuk menyetel *hyperparameter* kritis dari berbagai model ML, seperti Naive Bayes, *random forest*, *decision tree*, *XGboost*, dan *lightgbm*. Dataset yang digunakan adalah data BDHS (*Bangladesh Demographic and Health Survey*) dan performanya akurasinya diuji dengan metode AUC (*area under the receiver operating characteristic curve*).

Secara individual, lightgbm yang dioptimalkan memberikan presisi tertinggi dan AUC masing-masing sebesar 79,90% dan 77,80%. Hasil ini meningkat ketika lightgbm yang dioptimalkan dipadukan dengan *XGboost*, memberikan presisi dan AUC masing-masing sebesar 84,60 % dan 80,0 %. Hasil penelitian ini mengungkapkan bahwa teknik imputasi median statistik dengan metode pemilihan atribut berbasis *XGboost* dan pengklasifikasi *lightgbm* memberikan hasil individual terbaik. Performa meningkat ketika ansambel berbobot yang diusulkan dari pendekatan *XGboost* dan *lightgbm* diadaptasi dengan preprocessing yang sama dan direkomendasikan untuk penggunaan vaksin campak. Signifikansi dari pendekatan yang diusulkan penelitian ini adalah menggunakan atribut minimum yang dikumpulkan dari anak dan anggota keluarga mereka dan menghasilkan akurasi 80,0%, membuatnya mudah dijelaskan oleh pengasuh dan petugas kesehatan (Hasan et al., 2021).

Hasil *tuning hyperparameter* beberapa model ditunjukkan pada tabel 2.10 seperti berikut ini.

Table 2. 10 hasil tuning hyperparameter dan hasil uji AUC(Hasan et al., 2021)

|  |  |  |
| --- | --- | --- |
| **Model** | **Hasil tuning hyperparameter** | **AUC** |
| **GNB** | Variasi atribut terbesar untuk stabilitas (var\_smothing = 0.01) |  |
| **BNB** | Parameter halus laplace (alpha = 0.0), fitur binarizing threshold (binarize = 0.4), dan kelas prior (fit\_prior = true) |  |
| **RF** | Fungsi pembagi (criterion = gini), jumlah pembagian fitur (max\_features = auto), jumlah daun node (max\_leaf\_nodes = none), dan jumlah daun node sample (min\_sample\_leaf = 1.0) |  |
| **Decision Tree** | Fungsi pembagi (criterion = entrophy), kebijakan partisi node (splitter = best), split feature number (max\_feature = log2), jumlah internal node sample (min\_samples\_split = 0.1) dan jumlah daun sample node’s (min\_sample\_leaf = 0.05) |  |
| **XGBoost** | Jumlah hessian (min\_child\_weight = 1.0), loss minimum (gamma = 1.0), rasion instance subdsample (subsample = 0.25), rasio subsample kolom (colsample\_bytree = 1.0), kedalaman pohon (max\_depth = 5) |  |
| **LGBoost** | Base learner daun pohon (num\_leaves = 25), jumlah pohon (n\_estimators = 50), rasion subsample (subsample = 0.25), rasio subsample kolom (cosample\_bytree = 1.0), kedalaman pohon (max\_depth = -1) |  |

#### Model pembelajaran mesin yang efisien untuk prediksi kekuatan beton

Dalam penelitian ini, *machine learning* akan diimplementasikan untuk memprediksi kekuatan tekan dan tarik beton HPC *(High Performance Concrete).* Algoritma yang akan digunakan adalah GBR (*Gradient Boosting Regressor*),SVR (*Support Vector Regressor*), MLP (*MultiLayer Perceptron*), dam *XGBoost* (*eXtreme Gradient Boosting*). Metode *tuning* untuk *hyperparameter* yang digunakan adalah *Random Search.*

Dalam persiapan data, nilai yang hilang atau *missing value* akan diisi dengan rata-rata dari data yang tersedia yang memungkintakan lebih banyak informasi yang dapat digunakan dalam proses pelatihan. Hasil prediksi model menunjukkan bahwa model yang terbaik yang memiliki performa bagus adalah GBR dan *XGBoost* dibandingkan dengan model SVR dan MLP. Hasil *tuning hyperparameter* bisa dilihat pada tabel 2.11 dan 2.12(Nguyen et al., 2021).

Table 2. 11 perbandingan performa masing-masing model untuk dataset kekuatan tekan(Nguyen et al., 2021)

Table

Description automatically generated

Table 2. 12 perbandingan performa masing-masing model untuk dataset kekuatan tarik(Nguyen et al., 2021)

Graphical user interface, table

Description automatically generated

#### Prediksi Gagal Jantung menggunakan algoritma Xgboost dan seleksi fitur menggunakan permutasi fitur

Jantung adalah organ vital tubuh manusia yang menyediakan darah kaya oksigen bagi tubuh. Jika jantung berhenti bekerja, maka manusia akan mati dalam rentang waktu beberapa menit. Penelitian ini menganalisis kinerja algoritma *XGBoost* dalam prediksi kematian di antara pasien gagal jantung. Performa *XGBoost* dibandingkan dengan berbagai algoritma prediksi lainnya. *Tuning hyperparameter* telah dilakukan pada algoritma *XGBoost* untuk meningkatkan kinerja model dan mengurangi *overfitting*. Pada bagian kedua dari penelitian ini, pemilihan fitur menggunakan permutasi fitur telah dilakukan dengan menggunakan fitur penting dari dataset yang telah dipilih. Pencarian *hyperparameter* menggunakan metode *RandomSearchCV* dari *sklearn*. Lipatan bertingkat atau *stratified fold* digunakan selama *tuning hyperparameter* untuk memastikan bahwa data pelatihan dan pengujian tetap seimbang selama pembuatan dan prediksi model. Menggunakan pemilihan fitur, telah ditetapkan bahwa menggunakan sejumlah kecil fitur penting dapat meningkatkan kinerja model dengan jumlah yang signifikan dan dapat mengurangi *overfitting*.

Table 2. 13 hasil tuning hyperparameter optimal(Kaushik & Birok, 2021)

|  |  |  |
| --- | --- | --- |
| ***Hyperparameter*** | ***Optimal Value achieved*** | ***Description*** |
| *Minimum child weight* | 1 | Minimal weight yang dibutuhkan anak *tree* |
| *subsample* | 0.4 | Rasio subsample pelatihan |
| *gamma* | 3.5 | Parameter emangkasan leaf |
| *Colsample bytree* | 0.9 | Kolom subsample |
| *Maximum depth* | 3 | Maksimum keadalam pohon |
| *N estimators* | 125 | Jumlah pohon yang dibutuhkan |
| *Learning rate* | 0.15 | Parameter skala learning rate |

#### Prediksi pembelajaran mesin sifat magnetik gelas logam berbasis Fe dengan mempertimbangkan kemampuan pembentukan kaca

*Metallic Glass* berbasis Fe (MG) telah menunjukkan nilai komersial yang besar karena sifat magnetik lunaknya yang sangat baik. Prediksi magnetisme dengan pertimbangan kemampuan pembentukan kaca sangat penting untuk mengembangkan MG berbasis Fe. Namun, teori atau model yang dibuat berdasarkan fisika *condense matter* menunjukkan akurasi yang terbatas dan beberapa pengecualian. Dalam penelitian ini, berdasarkan 618 sampel MG berbasis Fe yang dikumpulkan dari penelitian yang yang telah dipublikasi, model *machine learning* (ML) dilatih dengan baik untuk memprediksi magnetisasi jenuh () dari MG berbasis Fe. *Glass forming ability* diperlakukan sebagai fitur menggunakan data eksperimen dari wilayah cairan superdingin (Tx). Tiga algoritma ML yang digunakan, yaitu ANN (*Artificial Neural Network*) , dan RF (*Random Forest*), *XGBoost* (*eXtreme Gradient Boosting*). Melalui pemilihan fitur dan *tuning hyperparameter*, *XGBoost* menunjukkan kinerja prediktif terbaik pada dataset uji yang dipisahkan secara acak dengan koefisien determinasi MAPE (*mean absolute percent error*) sebesar 5,563%, (R2) sebesar 0,942, dan RMSE (*root mean squared error*) sebesar 0,078 T. Berbagai peringkat kepentingan fitur yang diturunkan oleh model *XGBoost* menunjukkan bahwa Tx memainkan peran penting dalam performa prediksi model. Penelitian ini menunjukkan metode ML yang diusulkan dapat secara bersamaan menggabungkan GFA dan fitur lainnya dalam termodinamika, kinetika, dan struktur untuk memprediksi sifat magnetik MG berbasis Fe dengan akurasi yang sangat baik. Hasil *tuning hyperparameter* bisa dilihat pada gambar 2.11.

Chart

Description automatically generated

Gambar 2. 11 Hasil penyetelan hyperparameter untuk XGBoost dan RF dievaluasi oleh . Proses penyetelan (a) N estimator di XGBoost dan RF, (b) max depth dan minimum child weight di XGBoost, dan (c) maximum depth dan minimum samples split dari Random Forest. (d) Peringkat kepentingan fitur diperoleh dari model XGBoost terlatih. Skor kepentingan dinormalisasi dengan membagi skor maksimum, dan skor maksimum ditetapkan menjadi 100(X. Li et al., 2022).

#### Memprediksi Penipuan dalam Layanan Pembayaran Keuangan melalui Model XGBoost Hyper-Parameter-Tuned yang Dioptimalkan

Transaksi online, layanan medis, transaksi keuangan, dan perbankan semuanya memiliki andil dalam aktivitas penipuan. Pendapatan tahunan yang dihasilkan oleh penipuan melebihi $1 triliun. Meskipun penipuan berbahaya bagi organisasi, hal itu dapat diungkap dengan bantuan solusi cerdas seperti *rules engine* dan *machine learning*. Dalam penelitian ini, teknik hibrid unik digunakan untuk mengidentifikasi penipuan pembayaran finansial dengan menggabungkan *nature-inspired-based* *hyperparameter* dengan beberapa model *supervised learning*, seperti yang diterapkan dalam versi Algoritma *modified* *XGBoost*. Pertama, peneliti membagi sampel kumpulan data pembayaran keuangan lengkap untuk digunakan sebagai kumpulan pengujian. Data yang digunakan 70% data untuk pelatihan dan 30% untuk pengujian. Rekaman yang diketahui tidak sah atau palsu diprediksi, sedangkan yang menimbulkan kecurigaan diselidiki lebih lanjut menggunakan sejumlah algoritma pembelajaran mesin. Model dilatih dan divalidasi menggunakan teknik *10-fold cross-validation*.

Pada penelitian ini algoritma *nature-inspired-based hyperparemeter* yang diusulkan peneliti menyediakan kemampuan modifikasi pada model *XGBoost* yang meningkatkan akurasi dari model dalam menentukan mana saja yang merupakan penipuan. Penelitian ini menunjukkan performa yang cukup bagus dengan akurasi 99.68%, sedangkan *error rate* untuk memprediksi penipuan diharapkan sekitar 0.32%. Perbandingan dengan model lain bisa kita lihat pada tabel 2.14.

Table 2. 14 Perbandingan performa(Dalal et al., 2022)

|  |  |  |
| --- | --- | --- |
| **Algoritma** | **AUC** | **Akurasi** |
| Neural Network | 0.172 | 92.18 |
| Random Forest | 0.573 | 96.15 |
| Logistic Regression | 0.655 | 92.737 |
| Quest | 0.569 | 96.648 |
| Random Tree | 0.652 | 96.089 |
| Bayesian Network | 0.5 | 91.06 |
| Tree-AS | 0.681 | 96.648 |
| C5.0 | 0.5 | 96.648 |
| Discriminant algorithm | 0.667 | 97.207 |
| Modified XGBoost | 0.9988 | 99.68 |

## BAB 3 METODE PENELITIAN

### Alur Penelitian

Diagram dari alur penelitian dapat dilihat pada gambar 3.1

Diagram

Description automatically generated

Gambar 3. 1 Diagram alur penelitian

Dataset yang akan digunakan masih memilki nilai yang kosong, sehingga data perlu diproses terlebih dahulu. Pada proses ini nilai *missing value* atau nilai yang hilang akan diisi dengan nilai rata-rata antara 2 data sebelum dan sesudah, karena nilai yang hilang akan mempengaruhi pola historis dari dataset sehingga mempengaruhi proses pelatihan. Selanjutnya data yang sudah siap dipilih fitur yang akan digunakan dalam proses pelatihan. Fitur ini mencakup kelembapan udara, tekanan atmosfer, curah hujan, kecepatan angin, suhu udara, klasifikasi hujan atau tidak hujan untuk klasifikasi biner, klasifikasi hujan sangat ringan, hujan sedang & ringan, dan tidak hujan untuk klasifikasi multiclass. Selanjutnya data dibagi menjadi data *train* dan *test* dan dilakukan optimasi *hyperparameter* untuk mendapatkan *hyperparameter* terbaik untuk model *eXtreme Gradient Boosting.* Terakhir model dievaluasi sebarapa baik model dalam melakukan klasifikasi dan berapa skor *metrics* evaluasi lainnya.

### Tahapan Penelitian

Penelitian dilakukan dengan beberapa tahapan untuk mendapatkan hasil yang optimal. Tahapannya adalah sebagai berikut:

#### Idetifikasi Masalah

Latar belakang masalah dalam penelitian ini adalah banyaknya pengaruh yang ditimbulkan oleh besarnya curah hujan. Curah hujan sangat mempengaruhi kehidupan manusia mulai dari dampak positif maupun dampak negatif. Dalam mengatasi dampak negatif maka perlu adanya sistem yang dapat memperkirakan curah hujan yang akan terjadi. Oleh karena itu *eXtreme Gradient Boosting* diusulkan sebagai model untuk prediksi klasifikasi curah hujan.

#### Studi Literature

Kajian pustaka dilakukan berupa mengkaji literatur dan beberapa penelitian terkait yang sudah pernah dilakukan sebelumnya tentang topik yang akan dibahas. Kajian literatur yang dibahas adalah mengenai pembelajaran mesin, algoritma *decision tree,* dan *random forest* yang menjadi landasan dalam *gradient boosting,* lalu algoritma *eXtreme Gradient Boosting* dan penelitian terkait yang menggunakan model tersebut*.*

#### Data Collecting

Data yang digunakan adalah data curah hujan stasiun cuaca milik Badan Meteorologi Klimatologi dan Geofisika yang dikumpulkan menggunakan *Automated Weather Observed System* (AWOS) dalam kurun waktu tiga bulan dengan resolusi per jam untuk Kota Pontianak.

#### Data Understanding

Pada tahap ini data yang telah dikumpulkan dipahami secara keseluruhan mencakup informasi dalam data dan menentukan kualitasnya. Pemahaman data sangat penting untuk menghindari masalah yang tidak terduga selama tahapan berikutnya, yaitu tahap persiapan data (*data preparation*).

#### Data Preparation

Secara umum, hal-hal yang akan Anda lakukan dalam *data preparation* adalah sebagai berikut:

* Menggabungkan data.
* Menyeleksi data yang akan digunakan.
* Melakukan proses transformasi data
* Membagi data menjadi data training dan test.

#### Modeling dan Evaluation

Pemodelan dilakukan dalam beberapa iterasi. Pada tahap ini, peneliti menggunakan algoritma *XGBoost* untuk membuat model estimasi. Model ini akan dilatih dengan default *hyperparameter* dan melakukan evaluasi performa dengan *metrics:*

1. Akurasi: merupakan persentase dari prediksi benar yang dilakukan oleh model dari semua total prediksi. Dapat dituliskan dalam persamaan 3.1:
2. Presisi: Rasio *true* positif terhadap jumlah total prediksi positif yang dibuat oleh model. Dihitung dalam persamaan 3.2:
3. *Recall* (sensitivitas): Rasio *true* positif terhadap jumlah total positif actual dalam kumpulan data. Dihitung sebagai persamaan 3.3:
4. F1-score: Rata-rata *harmonic* dari presisi dan *recall*. Ini memberikan kepentingan yang sama untuk presisi dan *recall*. Ini dihitung sebagai persamaan 3.4:

#### Model Optimization (*Hyperparameter tuning*)

Tahap ini model akan di-*tuning* padatiga variasi *hyperparameter-*nyayaitu *max\_depth, learning\_rate,* dan *n\_estimator.* Pada penelitian ini fokus parameter yang akan dituning adalah untuk mencari nilai *learning\_rate* yang terbaik untuk model *eXtreme Gradient Boosting* dan parameter lainnya dibiarkan dengan nilai default*.*

### Tempat Penelitian

Penelitian ini dilakukan di Laboratorium Cisco, Departement Fisika, FMIPA Universitas Indonesia, Kota Depok.

### Jadwal Penelitian

Berikut ini adalah agenda penelitian dan jadwal pelaksanannya:

Tabel 3. 1 Tabel agenda dan jadwal penelitian

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kegiatan** | **2023** | | | | | | |
| **Bulan ke-** | | | | | | |
| **2** | **3** | **4** | **5** | **6** | **7** |  |
| Tahap I (persiapan) | | | | | | | | |
| 1 | Kajian Literatur |  |  |  |  |  |  |  |
| 2 | Penyusunan Proposal |  |  |  |  |  |  |  |
| Tahap II (Penelitian) | | | | | | | | |
| 3 | Pengolahan data |  |  |  |  |  |  |  |
| 4 | Eksplorasi data |  |  |  |  |  |  |  |
| 5 | Analisis data |  |  |  |  |  |  |  |
| 6 | Hyperparameter Optimization |  |  |  |  |  |  |  |
| Tahap II (Penulisan) | | | | | | | | |
| 7 | Pembuatan naskah skripsi |  |  |  |  |  |  |  |
| 8 | Sidang skripsi |  |  |  |  |  |  |  |
| 9 | Revisi Skripsi dan pembuatan manuskrip makalah ilmiah |  |  |  |  |  |  |  |

### Kesimpulan

Metode yang akan digunakan dalam penelitian ini adalah *eXtreme Gradient Boosting* untuk premasalah klasifikasi. Harapannya penelitian ini dapat diselesaikan dengan hasil estimasi curah hujan yang memiliki presisi yang baik, dan dapat diterapkan dalam melakukan estimasi curah hujan di instansi terkait.

## BAB 4 HASIL DAN PEMBAHASAN

### *Data Understanding*

Dalam masalah klasifikasi, algoritma pembelajaran mesin mempelajari pola dari dataset yang diberikan (Navas, 2022). Dataset tersebut berupa variable yang dapat mempengaruhi variable target yang akan dilakukan estimasi atau prediksi. Penelitian ini melakukan prediksi terhadap data kategori curah hujan, atau klasifikasi curah hujan, dimana algoritma melakukan estimasi terhadap data kategori *binary classification* dan *multiclass classification*. Sehingga ada beberapa faktor fisis yang dapat digunakan untuk melakukan estimasi curah hujan, seperti:

1. Kelembaban udara: Tingginya kelembaban udara dapat meningkatkan peluang terjadinya hujan karena udara yang basah bisa menampung lebih banyak uap air.
2. Suhu: Penurunan suhu dapat menyebabkan terjadinya kondensasi uap air di udara yang akhirnya membentuk awan dan hujan.
3. Tekanan atmosfer: Tekanan atmosfer yang rendah bisa menyebabkan konveksi udara dan mengakibatkan pembentukan awan hujan.
4. Kecepatan angin: Kecepatan angin yang tinggi bisa membawa uap air dari jarak jauh dan memicu pembentukan awan hujan (Yu et al., 2022).

Pada penelitian ini, variabel atau dataset diambil pada stasiun cuaca kota Pontianak, dengan menggunakan alat *Automated Weather Observed System* (AWOS). Data yang dikumpulkan adalah data per jam, pada tanggal 1 Desember 2022 hingga 23 Februari 2023. Data pada penelitian ini berfokus pada faktor-faktor fisis apa saja yang mungkin dapat mempengaruhi terjadinya hujan. Untuk dapat melakukan estimasi ataupun prediksi terkait permasalahan klasifikasi curah hujan dibutuhkan data input dan data output atau target. Data yang dikumpulkan adalah data suhu udara, kelembapan udara, tekanan atmosfer, arah angin megnetik, arah angin sejati, kecepatan angin, dan curah hujan. Data secara keseluruhan dapat dilihat pada table 4.1

Tabel 4. 1 Tabel dataset yang digunakan dalam penelitian.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Date and Time** | **Air Tmp (C) M 60 Min** | **Mag WD 60 Min (deg) M** | **Precip 1Hr (mm) M** | **QNH (hPa) M** | **RH (%) M 60 Min** | **True WD 60 Min (deg) M** | **WS 60 Min (kt) M** |
| 0 | 01/12/2022 00:00 | 23.73 | 82.0 | 29.7 | 1009.01 | 100.0 | 84.0 | 2.0 |
| 1 | 01/12/2022 01:00 | 26.51 | 61.0 | 29.5 | 1009.35 | 92.9 | 63.0 | 2.0 |
| 2 | 01/12/2022 02:00 | 28.45 | 3.0 | 23.5 | 1009.47 | 81.8 | 5.0 | 2.0 |
| 3 | 01/12/2022 03:00 | 29.59 | 340.0 | 21.9 | 1009.11 | 75.2 | 342.0 | 2.0 |
| 4 | 01/12/2022 04:00 | 30.68 | 297.0 | 19.9 | 1008.14 | 69.8 | 299.0 | 4.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2035 | 23/02/2023 19:00 | 25.00 | 269.0 | NaN | 1008.60 | 90.0 | 271.0 | 2.0 |
| 2036 | 23/02/2023 20:00 | 24.70 | 301.0 | NaN | 1008.37 | 92.2 | 303.0 | 2.0 |
| 2037 | 23/02/2023 21:00 | 24.31 | 312.0 | NaN | 1008.40 | 92.9 | 314.0 | 2.0 |
| 2038 | 23/02/2023 22:00 | 24.43 | 334.0 | NaN | 1008.74 | 92.0 | 336.0 | 4.0 |
| 2039 | 23/02/2023 23:00 | 24.16 | 331.0 | NaN | 1009.32 | 92.5 | 333.0 | 3.0 |

Berdasarkan data table 4.1 bisa dilihat informasi statistiknya untuk mengetahui total data (count), nilai rata-rata (mean) dari masing-masing data, standard deviasi (std) atau simpangan data, nilai minimum (min) dari dataset, quartil 1 (25%), quartil 2 (50%), quartil 3 (75%) dan nilai maximum (max) dari dataset penelitian. Secara rinci ditampilkan pada tabel 4.2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Air Tmp (C) M 60 Min** | **Mag WD 60 Min (deg) M** | **Precip 1Hr (mm) M** | **QNH (hPa) M** | **RH (%) M 60 Min** | **True WD 60 Min (deg) M** | **WS 60 Min (kt) M** |
| count | 1.905 | 1.906 | 1.918 | 1.917 | 1.906 | 1.906 | 1.906 |
| mean | 26.35 | 204.62 | 0.27 | 1.009.29 | 84.79 | 203.34 | 3.75 |
| std | 2.81 | 118.34 | 1.82 | 2.000.46 | 13.23 | 118.87 | 2.43 |
| min | 21.92 | 2.00 | 0.00 | 1.002.98 | 43.30 | 1.00 | 0.00 |
| 25% | 24.10 | 78.00 | 0.00 | 1.008.00 | 74.92 | 78.00 | 2.00 |
| 50% | 25.40 | 247.50 | 0.00 | 1.009.35 | 89.40 | 245.00 | 3.00 |
| 75% | 28.64 | 305.75 | 0.00 | 1.010,67 | 96.00 | 305.00 | 5.00 |
| max | 33.71 | 360.00 | 29.70 | 1.014,26 | 100.00 | 360.00 | 13.00 |

Tabel 4. 2 Statistik Dataset

Dari nilai count pada tabel 4.2 dapat dilihat bahwa total data tidak sama untuk masing-masing fitur. Total data pada tabel 4.1 adalah 2040, sedangkan total yang tercatat pada tabel 4.2 hanya berkisar 1900, sehingga hal ini menunjukkan adanya nilai yang hilang (*missing value*), dan nilai *missing value* ini akan diselesaikan pada tahap persiapan data.

Pada tabel 4.2 terdapat 7 variabel, yaitu Date and Time, Air tmp (C) M 60 min, Mag WD 60 Min (deg) M**,** Precip 1Hr (mm) M, QNH (hPa) M, RH (%) M 60 Min, True WD 60 Min (deg) M, WS 60 Min (kt) M. Secara rinci penjelasan variable dan nilai missing valuenya ditampilkan pada table 4.3.

Tabel 4. 3 Penjelasan masing-masing variable dan nilai yang hilang

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Fitur | Deskripsi | Total data yang hilang | Satuan |
| 1 | Date and Time | Tanggal dan waktu perjamnya | 135 | Jam |
| 2 | Air tmp (C) M 60 min | Suhu udara | 134 | Celsius |
| 3 | Mag WD 60 Min (deg) | Arah angin magnetik | 122 | Degre |
| 4 | Precip 1Hr (mm) M | Curah hujan | 123 | Milimeter |
| 5 | QNH (hPa) M | Tekanan Atmosfer | 134 | Hektopascal |
| 6 | True WD 60 Min (deg) | Arah angin sejati | 134 | Degre |
| 7 | WS 60 Min (kt) M | Kecepatan Angin | 134 | Knot |

Fitur target untuk masalah klasifikasi adalah data kategori, oleh karena itu untuk klasifikasi curah hujan data akan dikelompokkan menjadi *binary classification* dan *multiclass classification* dan dilakukan evaluasi terhadap klasifikasi tersebut dan dipilih metode klasifikasi yang mendukung tujuan dan manfaat penelitian dengan metrik evaluasi terbaik. Pertama untuk *binary classification,* data curah hujan akan dikelompokkan menjadi hujan ketika data curah hujan lebih besar dari 0 dan tidak hujan ketika data curah hujan sama dengan 0. Sedangkan *multiclass classification* akan dikelompokkan sesuai dengan standar international World Meteorogical Organization (WMO) seperti ditampilkan pada tabel 4.4.

Tabel 4. 4 Tabel klasifikasi data curah hujan berdasarkan World Meteorogical Organization (WMO)

|  |  |  |
| --- | --- | --- |
| **Kriteria Hujan** | **Curah hujan** | **Jumlah** |
| Sangat ringan | < 5.0 mm | 179 |
| Ringan | 5.0 – 20 mm | 23 |
| Sedang / Normal | 20 – 50 mm | 4 |
| Lebat | 50 – 100 mm | 0 |
| Sangat Lebat | >100 mm | 0 |
| Tidak hujan\* | 0 mm | 1834 |
| \*tambahan diluar klasifikasi WMO | | |

Tabel 4.4 menunjukkan total masing-masing klasifikasi curah hujan. Kategori lebat memiliki total data 0 dan sangat lebat juga 0, maka dua kategori ini akan dihapuskan. Sedangkan kategori hujan sedang bernilai terlalu kecil yaitu 4, oleh karena itu kategori ini akan digabungkan dengan kategori ringan dan membuat kategori baru yaitu kategori ringan dan sedang seperti pada tabel 4.5

Tabel 4. 5 Tabel klasifikasi data secara keseluruhan

|  |  |  |
| --- | --- | --- |
| **Kriteria Hujan** | **Multiclass classification** | **Binary classification** |
| Sangat ringan | 179 | 206 |
| Ringan dan sedang | 27 |
| Tidak hujan | 1834 | 1834 |

Dataset secara keseluruhan ditampilkan pada tabel 4.6

Tabel 4. 6 Keseluruhan dataset

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Date and Time** | **Air Tmp (C) M 60 Min** | **Precip 1Hr (mm) M** | **QNH (hPa) M** | **RH (%) M 60 Min** | **WS 60 Min (kt) M** | **multiclass classification** | **Binary Classification** |
| 0 | 01/12/2022 00:00 | 23.73 | 0.0 | 1009.01 | 100.0 | 2.0 | tidak hujan | tidak hujan |
| 1 | 01/12/2022 01:00 | 26.51 | 0.0 | 1009.35 | 92.9 | 2.0 | tidak hujan | tidak hujan |
| 2 | 01/12/2022 02:00 | 28.45 | 0.0 | 1009.47 | 81.8 | 2.0 | tidak hujan | tidak hujan |
| 3 | 01/12/2022 03:00 | 29.59 | 0.0 | 1009.11 | 75.2 | 2.0 | tidak hujan | tidak hujan |
| 4 | 01/12/2022 04:00 | 30.68 | 0.0 | 1008.14 | 69.8 | 4.0 | tidak hujan | tidak hujan |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2035 | 23/02/2023 19:00 | 25.00 | NaN | 1008.60 | 90.0 | 2.0 | tidak hujan | tidak hujan |
| 2036 | 23/02/2023 20:00 | 24.70 | NaN | 1008.37 | 92.2 | 2.0 | tidak hujan | tidak hujan |
| 2037 | 23/02/2023 21:00 | 24.31 | NaN | 1008.40 | 92.9 | 2.0 | tidak hujan | tidak hujan |
| 2038 | 23/02/2023 22:00 | 24.43 | NaN | 1008.74 | 92.0 | 4.0 | tidak hujan | tidak hujan |
| 2039 | 23/02/2023 23:00 | 24.16 | NaN | 1009.32 | 92.5 | 3.0 | tidak hujan | tidak hujan |

### *Data Preparation*

Data yang telah dikumpulkan akan dilakukan persiapan agar dapat dipahami oleh model. Pada proses ini akan dilakukan penganganan terhadap data yang hilang atau missing value, pembagian dataset menjadi data latih dan uji, standarisasi data latih dan data uji.

#### Penanganan *missing value*

Data yang hilang dapat mempengaruhi proses pelatihan. Terdapat beberapa metode yang digunakan untuk menangani missing value, diantaranya adalah menghapus nilai yang hilang tersebut atau dengan mengambil nilai rata-rata kolom tersebut dan mengisinya pada bagian missing value. Pada penelitian ini data yang hilang akan diganti dengan nilai rata-rata agar dapat mempertahankan data hostorisnya. Nilai rata-rata yang diambil adalah nilai rata-rata dua data yaitu nilai sebelum dan setelah *missing value.* Seperti yang ditampilkan pada gambar 4.1

Table

Description automatically generated

Gambar 4. 1 Ilustrasi penaganan missing value

Nilai *missing value* pada tabel tersebut akan mengambil nilai rata-rata dari nilai 22.49 dan 24.77, sehingga nilai pada missing value tidak terlalu jauh melenceng dari data yang sebenarnya, karena resolusi data adalah perjam sehingga data satu jam setelahnya atau beberapa jam setelahnya kemungkinan besar tidak sejauh nilai sebelum dan sesudah *missing value* tersebut.

#### One-hot Encoding

Dalam pembelajaran mesin data kategori yang berbentuk teks atau tipe data string tidak bisa diproses oleh model. Oleh karena itu data kategori ini perlu diubah kedalam tipe data yang dapat dikenali dan bisa diproses oleh model, yaitu menjadi data numerik (Dahouda & Joe, 2021). Dalam hal ini, salah satu metode yang dapat digunakan adalah *one-hot encoding.* Cara kerja dari metode ini adalah dengan mengubah data kategori menjadi bit vector, dengan nilai 0 dan 1. Setiap bit vector ini merepresentasikan satu kemungkinan nilai, artinya panjang dari vector ini sama dengan jumlah dari kemungkinan nilai atau jumlah kategori yang ada (Erjavac et al., 2022). Hasil *encoding* fitur kategori dapat dilihat pada tabel 4.7.

Tabel 4. 7 *one-hot encoding* fitur kategori pada dataset

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Binary classification** | | | | | | | | |
| **Date and Time** | **Air Tmp (C) M 60 Min** | **Precip 1Hr (mm) M** | **QNH (hPa) M** | **RH (%) M 60 Min** | **WS 60 Min (kt) M** | **Klasifikasi hujan** | **Klasifikasi tidak hujan** | |
| 01/12/2022 00:00 | 23.73 | 0.0 | 1009.01 | 100.0 | 2.0 | 0 | 1 | |
| 01/12/2022 01:00 | 26.51 | 0.0 | 1009.35 | 92.9 | 2.0 | 0 | 1 | |
| 01/12/2022 02:00 | 28.45 | 0.0 | 1009.47 | 81.8 | 2.0 | 0 | 1 | |
| 01/12/2022 03:00 | 29.59 | 0.0 | 1009.11 | 75.2 | 2.0 | 0 | 1 | |
| 01/12/2022 04:00 | 30.68 | 0.0 | 1008.14 | 69.8 | 4.0 | 0 | 1 | |
| ... | ... | ... | ... | ... | ... | ... | ... | |
| **Multiclass classification** | | | | | | | | |
| **Date and Time** | **Air Tmp (C) M 60 Min** | **Precip 1Hr (mm) M** | **QNH (hPa) M** | **RH (%) M 60 Min** | **WS 60 Min (kt) M** | **Klasifikasi ringan dan sedang** | **Klasifikasi sangat ringan** | **Klasifikasi tidak hujan** |
| 01/12/2022 00:00 | 23.73 | 0.0 | 1009.01 | 100.0 | 2.0 | 0 | 0 | 1 |
| 01/12/2022 01:00 | 26.51 | 0.0 | 1009.35 | 92.9 | 2.0 | 0 | 0 | 1 |
| 01/12/2022 02:00 | 28.45 | 0.0 | 1009.47 | 81.8 | 2.0 | 0 | 0 | 1 |
| 01/12/2022 03:00 | 29.59 | 0.0 | 1009.11 | 75.2 | 2.0 | 0 | 0 | 1 |
| 01/12/2022 04:00 | 30.68 | 0.0 | 1008.14 | 69.8 | 4.0 | 0 | 0 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

#### Pemilihan Fitur

Data fitur independent atau input yang akan digunakan hanya 4 poin yang dijelaskan pada data understanding yaitu, suhu udara, kelembapan udara, tekanan atmosfer dan kecepatan angin, sedangkan arah angin tidak digunakan karena hanya berpengaruh terhadap lokasi kejadian hujan. Sedangkan fitur dependen atau target adalah data kategori yang sudah kita lakukan *one-hot encoding,* yaitu klasifikasi hujan dan klasifikasi tidak hujan untuk *binary classification,* lalu klasifikasi ringan sedang, klasifikasi sangat ringan, dan klasifikasi tidak hujan untuk *multiclass classification*.

#### Pembagian dataset

Dalam proyek ini, dataset yang ada perlu dibagi menjadi dua bagian yaitu data train dan data test. Data train digunakan untuk melatih model, sementara data test belum diketahui oleh model dan digunakan untuk menguji model yang telah dirancang. Untuk proporsi pembagian dataset akan divariasikan dengan 90%, 80%, 70%, dan 60% untuk data latih, sedangkan sisanya untuk data uji yaitu 10%, 20%, 30%, dan 40%. Setelah dilakukan evaluasi akan dipilih proporsi pembagian yang memiliki performa terbaik. Untuk melakukan pembagian dataset tersebut, digunakan modul train\_test\_split dari scikit-learn. Sehingga, jumlah sampel yang digunakan pada model dapat dilihat pada table 4.8.

Tabel 4. 8 Tabel pembagian dataset *binary classification* dan *multiclass classification*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Kategori | Binary | | | | | | | | |
| 90%-10% | | 80%-20% | | 70%-30% | | 60%-40% | | |
| train | test | train | test | train | test | train | Test | |
| hujan | 187 | 19 | 174 | 32 | 153 | 53 | 128 | 78 | |
| tidak hujan | 1649 | 185 | 1458 | 376 | 1275 | 559 | 1096 | 738 | |
|  | | | | | | | | | |
| Kategori | Multiclass | | | | | | | |
| 90%-10% | | 80%-20% | | 70%-30% | | 60%-40% | |
| train | test | train | test | train | test | train | test |
| ringan sedang | 24 | 3 | 23 | 4 | 20 | 7 | 18 | 9 |
| sangat ringan | 163 | 16 | 151 | 28 | 133 | 46 | 110 | 69 |
| tidak hujan | 1649 | 185 | 1458 | 376 | 1275 | 559 | 1096 | 738 |

#### Standarisasi dataset

Dalam pelatihan model, proses standardisasi sangat penting karena model lebih mudah memproses data yang seragam dan mendekati distribusi normal. Karena data yang digunakan adalah fitur numerik, teknik yang digunakan adalah StandarScaler yang tersedia di library Scikitlearn. StandarScaler digunakan untuk melakukan standarisasi dengan mengurangi nilai rata-rata kemudian membaginya dengan standar deviasi untuk menggeser distribusi. Dalam proses ini, StandarScaler menghasilkan distribusi dengan standar deviasi yang sama dengan 1 dan mean yang sama dengan 0. Sebanyak 68% nilai akan berada di antara rentang -1 dan 1 (scikit learn Developers, 2023). Standarisasi dilakukan terlebih dahulu pada data latih untuk menghindari kebocoran data, sedangkan untuk data uji, standardisasi dilakukan pada tahap evaluasi. Salah satu distribusi standarisasi data train bisa dilihat pada tabel 4.9.

Tabel 4. 9 Tabel distribusi pada data train.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| count | 1836.00 | 1836.00 | 1836.00 | 1836.00 |
| mean | 0.00 | 0.00 | 0.00 | 0.00 |
| std | 1.00 | 1.00 | 1.00 | 1.00 |
| min | -1.62 | -3.20 | -3.18 | -1.56 |
| 25% | -0.81 | -0.63 | -0.73 | -0.71 |
| 50% | -0.31 | -0.03 | 0.32 | -0.29 |
| 75% | 0.78 | 0.69 | 0.85 | 0.55 |
| max | 2.63 | 2.57 | 1.17 | 3.92 |

### Implementasi Model *eXtreme Gradient Boosting*

Model *eXtreme Gradient Boosting* (XGBoost) merupakan model yang melakukan pembelajaran dengan membangun pohon keputusan dari data latih. Untuk mengontrol pohon keputusan agar dapat menghasilkan model dengan akurasi yang kuat, terdapat hyperparameter yang akan mengurangi *loss function*. Ada 3 hyperparameter utama yang digunakan, yaitu *max* *depth*, *learning* *rate*, dan *n-estimator.* Pada tahap modelling, model akan dilatih dengan default hyperparameter terlebih dahulu sebagai pembanding apakah model sudah optimal atau belum.

#### Default Hyperparamter

Untuk tahap pertama, nilai hyperparameter menyesuaikan dengan penelitian yang sudah dilakukan sebelumnya mengenai prediksi curah hujan oleh Anwar et al tahun 2021 yang menggunakan nilai default hyperparameter untuk XGBoost, seperti pada tabel 4.10. Nilai ini akan diimplementasikan pada fungsi *XGBClassifier* yang terdapat dalam modul *xgboost* untuk menangani klasifikasi*.* Secara rinci bisa dilihat dalam gambar 4.2.

Tabel 4. 10 penjelasan hyperparameter dan penetapan nilai awalnya

|  |  |  |
| --- | --- | --- |
| Hyperparemeter | Keterangan | Nilai |
| *Max depth* | Nilai yang mengontrol kedalaman pohon dari node root hingga node terjauh | 6 |
| *Learning rate* | Nilai yang mengontrol seberapa cepat model melakukan pembelajaran | 0.3 |
| *n-estimator* | Nilai yang mengatur banyak pohon keputusan yang akan dibangun | 100 |

Text

Description automatically generated

Gambar 4. 2 Gambar model *XGBClassifier* secara default

#### Tuning hyperparameter

Hyperparameter yang akan dicari pada penelitian ini adalah nilai learning rate. Pencarian mengunakan teknik *coarse to fine* dengan *Randomized Search Cross Validation* dan *Grid Search Cross Validation* dari modul *sklearn.* Metode yang digunakan untuk *coarse search* (pencarian kasar) adalah *Random Search* yang mencari hyperparameter secara acak dalam ruang yang luas. Sedangkan *fine search* (pencarian halus) menggunakan *Grid Search* dengan menggunakan ruang pencarian di sekitar hasil yang diperoleh pada *coarse search* secara halus.

Text

Description automatically generated

Gambar 4. 3 Inisialisai rentang pencarian hyperparameter

Nilai learning rate pada pencarian secara kasar adalah dalam rentang 0,01 hingga 1, secara seragam. Sedangkan nilai n\_estimators dan max\_depth ditetapkan dengan nilai tetap yaitu 100 dan 6. Selanjutnya model dilatih dengan modul RandomSearchCV dari modul *sklearn.model\_selection*. Terdapat 6 parameter yang ditetapkan pada modul RandomSearchCV, yaitu estimator, param\_distributions, n\_iter, cv, scoring, dan n\_jobs. Penjelasan masing-masing parameter dapat dilihat pada tabel 4.11, dan implementasinya pada gambar 4.4.

Tabel 4. 11 penjelasan parameter yang digunakan pada modul RandomSearchCV

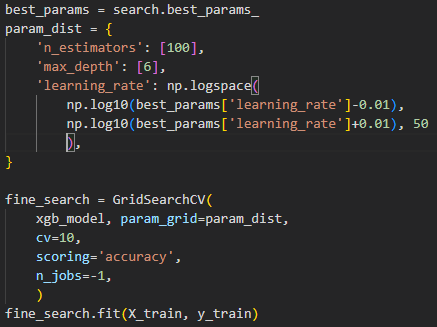
|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Keterangan | Nilai yang digunakan |
| 1. | estimator | Objek estimator yang akan dioptimasi, estimator harus dapat mengimplementasikan metode fit dan predict | xgb\_model |
| 2. | Param distributions | Dictonary dari parameter yang akan dioptimasi | Pada gambar 4.3 |
| 3. | n\_iter | Jumlah iterasi pencarian parameter acak | 50 |
| 4. | cv | Skema cross-validation yang akan digunakan dalam memvalidasi model. | 10 |
| 5. | Scoring | Metrik evaluasi yang akan digunakan untuk melakukan pencarian hyperparameter terbaik berdasarkan performa model | Accuracy |
| 6. | n\_jobs | Jumlah perkerjaan processor yang dilakukan secara parallel. | -1 yaitu menggunakan seluruh processor |

Text

Description automatically generated

Gambar 4. 4 Implementasi kode pada RandomSearchCV

Nilai hyperparameter optimal learning rate diambil dari *search.best\_params\_* dari modul *sklearn*, hasilnya adalah 0.02310 dan nilai ini disimpan dalam variable best\_params. Setelah pencarian dengan kasar selesai menggunakan RandomSearchCV, selanjutnya hasil learning rate digunakan sebagai ruang pencarian modul GridSearchCV dari *sklearn.modul\_selection*. Ruang pencarian dengan *grid search* adalah nilai best\_params dikurangi 0.01 hingga best\_params ditambah 0.01. Secara jelas implementasinya bisa dilihat pada gambar 4.5.



Gambar 4. 5 Implementasi GridSearchCV

### Hasil Tuning Hyperparameter

Pada Peneltian ini nilai hyperparameter yang dicari adalah nilai learning rate dengan menggunakan teknik coarse to fine. Pencarian hyperparameter dilakukan dengan menganalisis masing-masing pembagian data train dan data test. Hasil yang terbaik akan dipertimbangkan berdasarkan nilai akurasi pada data latih. Hasil tuning dapat dilihat pada tabel 4.15.

Tabel 4. 14 Hasil tuning hyperparameter

|  |  |  |
| --- | --- | --- |
| Split train-test | *Binary classification* | |
| Best learning rate | *Accuracy* |
| 90%-10% | 0.026 | 90.14% |
| 80%-20% | 0.047 | 90.25% |
| 70%-30% | 0.089 | 89.15% |
| 60%-40% | 0.024 | 89.62% |
|  | | |
| Split train-test | *Multiclass classification* | |
| Best learning rate | *Accuracy* |
| 90%-10% | 0.030 | 89.8% |
| 80%-20% | 0.042 | 89.2% |
| 70%-30% | 0.029 | 88.5% |
| 60%-40% | 0.036 | 88.8% |

Pada Tabel 4.14 dapat dilihat bahwa nilai *accuracy* data latih tertinggi untuk *binary* *classification* adalah pada split train dan test sebesar 90%-10%, sebesar 90.25% dan learning rate terbaik adalah 0.042. Sedangkan pada klasifikasi multiclass *accuracy* yang diperoleh sebesar 89.8% dan nilai learning rate sebesar 0.030. Nilai learning rate ini diperoleh berdasarkan model yang dibangun dari dataset, sehingga nilai ini bersifat *dependent* atau bergantung pada pohon keputusan masing-masing model. Oleh karena itu dengan nilai learning rate yang sama atau mendekati sama seperti pada pembagian klasifikasi binary 90%-10% dan 70%-30% yaitu, 0.026 dan 0.024 memiliki nilai akurasi yang cukup berbeda. Hal ini disebabkan karena proporsi dataset pada pembagian 90%-10% lebih besar dibanding 70%-30%, sehingga model yang dibangun lebih kompleks dan detail, dan membuat akurasi dari model lebih tinggi.

Hasil pencarian learning rate terbaik dapat divisualisasikan pada grafik seperti pada gambar 4.6.

|  |  |
| --- | --- |
| a)A picture containing screenshot, text, diagram, plot  Description automatically generated | b)A graph with red and blue dots  Description automatically generated with low confidence |
| c)A picture containing screenshot, text, diagram, plot  Description automatically generated | d) A graph with red and blue dots  Description automatically generated with low confidence |
| e)A picture containing text, screenshot, diagram, plot  Description automatically generated | f)  A graph with red and blue dots  Description automatically generated with low confidence |
| g)A graph with red and blue dots  Description automatically generated with low confidence | h)  A picture containing text, screenshot, diagram, plot  Description automatically generated |

Gambar 4. 6 Visualisasi pencarian learning rate dengan teknik coarse to fine. a) binary class 60%-40%, b) binary class 70%-30%, c) binary class 80%-20%, d) binary class 90%-10%, e) multiclass 60%-40%, f) multiclass 70%-30%, g) multiclass 80%-20%, h) multiclass 90%-10%

Bedasarkan gambar 4.6, dapat diperhatikan bahwa nilai learning rate yang memiliki akurasi tinggi berada pada rentan 0,0 hingga 0,1. Selain itu, pada dataset penelitian ini nilai akurasi berbanding terbalik dengan nilai learning rate, semakin tinggi learning rate akurasi semakin rendah. Hal ini menunjukkan bahwa learning rate yang kecil memiliki akurasi lebih besar dibandingkan dengan learning rate yang besar.

### Evaluasi Model

Setelah proses pelatihan selesai dilakukan selanjutnya model akan diuji menggunakan data uji. Data uji harus dilakukan proses standarisasi terlebih dahulu agar sama dengan data latih. Selanjutnya model diuji menggunakan metrik evaluasi dari modul *sklearn.metrics*, yaitu *accuracy\_score*, *precision\_score*, *recall\_score*, dan *f1\_score*. Hasil evaluasinya model bisa dilihat pada tabel 4.13 dan 4.14

Tabel 4. 12 Tabel evaluasi binary classification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matrik evaluasi | Default Hyperparameter | | | |
| Binary | | | |
| 90%-10% | 80%-20% | 70%-30% | 60%-40% |
| Accuracy | 91% | 91% | 90% | 88% |
| Precision | 89% | 90% | 87% | 86% |
| Racall | 91% | 91% | 90% | 88% |
| F1-score | 90% | 90% | 88% | 87% |
|  | | | | |
| Matrik evaluasi | Optimized Hyperparameter | | | |
| Binary | | | |
| 90%-10% | 80%-20% | 70%-30% | 60%-40% |
| Accuracy | 92% | 92% | 91% | 90% |
| Precision | 91% | 90% | 89% | 87% |
| Racall | 92% | 92% | 91% | 90% |
| F1-score | 90% | 90% | 90% | 88% |

Tabel 4. 13 Tabel evaluasi multiclass classification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matrik evaluasi | Default Hyperparameter | | | |
| Multiclass | | | |
| 90%-10% | 80%-20% | 70%-30% | 60%-40% |
| Accuracy | 89% | 89% | 89% | 87% |
| Precision | 89% | 89% | 87% | 86% |
| Racall | 90% | 90% | 90% | 88% |
| F1-score | 89% | 89% | 88% | 87% |
|  | | | | |
| Matrik evaluasi | Optimized Hyperparameter | | | |
| Multiclass | | | |
| 90%-10% | 80%-20% | 70%-30% | 60%-40% |
| Accuracy | 91% | 91% | 91% | 89% |
| Precision | 91% | 89% | 90% | 87% |
| Racall | 92% | 92% | 91% | 90% |
| F1-score | 90% | 90% | 89% | 87% |

Dari tabel 4.13 dan 4.14 dapat dilihat masing masing metrik evaluasi mengalami peningkatan yang signifikan sebesar 1% hingga 2% pada hyperparameter yang dituning. Pada tabel binary, model dengan split data train dan data test sebesar 90%-10% yang telah optimal memiliki nilai evaluasi tertinggi yaitu, *accuracy* 92%, *precision* 91%, *recall* 92%, *f1-score* 90%. Sedangkan pada klasifikasi multiclass nilai terbaik adalah pada pembagian *train*-*test* sebesar 90%-10% yaitu, *accuracy* 91%, *precision* 91%, *recall* 92%, *f1-score* 90%.

## BAB 5 KESIMPULAN DAN SARAN

### Kesimpulan

Berdasarkan percobaan yang telah dilakukan dapat ditarik kesimpulan sebagai berikut:

1. Rancangan metode klasifiksai curah hujan dengan menggunakan model *eXtreme Gradient Boosting* telah berhasil dilakukan dengan metrik evaluasi yang cukup bagus, yaitu diatas 80%.
2. Hasil pencarian learning rate dengan akurasi terbaik adalah pada kasus klasifikasi biner dengan pembagian data latih dan test 80:20 yaitu sebesar 0,047 dan akurasi sebesar 90.25%.
3. Learning rate bergantung pada dataset yang dilatih dalam membuat pohon keputusan, sehingga pada proporsi dataset yang berbeda, nilai ini akan menghasilkan akurasi yang berbeda meskipun dengan learning rate yang sama.
4. Learning rate yang kecil lebih akurat dalam melakukan prediksi dibandingkan dengan learning rate besar, akan tetapi pembelajaran model akan relatif lebih lama.
5. Model dengan data latih lebih besar dapat meninggkatkan akurasi model dalam melakukan prediksi, karena model memiliki pemahaman lebih baik dengan data yang banyak dibandingkan model dengan data latih yang sedikit.

### Saran

Pada penelitian ini masih terdapat beberapa hal yang perlu ditingkatkan diantaranya adalah sebagai berikut:

1. Dataset yang imbalance pada model dapat diatasi dengan metode undersampling, oversampling, atau metode penanganan data imbalance lainnya. Hal ini dapat memperbagus model dalam memahami dataset.
2. Dataset yang digunakan dapat diperbanyak, dan pengembangan model dapat menggunakan perangkat yang lebih bagus dan lebih cepat dalam pelatihan model.

## DAFTAR PUSTAKA

Agata, R., & Jaya, I. G. N. M. (2019). A comparison of extreme gradient boosting, SARIMA, exponential smoothing, and neural network models for forecasting rainfall data. *Journal of Physics: Conference Series*, *1397*(1). https://doi.org/10.1088/1742-6596/1397/1/012073

Anwar, M. T., Winarno, E., Hadikurniawati, W., & Novita, M. (2021). Rainfall prediction using Extreme Gradient Boosting. *Journal of Physics: Conference Series*, *1869*(1). https://doi.org/10.1088/1742-6596/1869/1/012078

Ciaburro, G., Ayyadevara, V. K., & Perrier, A. (2018). *Hands-on machine learning on Google cloud platform : implementing smart and efficient analytics using Cloud ML Engine* (S. Shetty, T. Gupta, C. Dsa, D. Pawar, V. Phadkay, N. Joshi, Safis, M. Chettiyar, & T. Dutta, Eds.; 1st ed., Vol. 1). Packt Publishing Ltd.

Dahouda, M. K., & Joe, I. (2021). A Deep-Learned Embedding Technique for Categorical Features Encoding. *IEEE Access*, *9*, 114381–114391. https://doi.org/10.1109/ACCESS.2021.3104357

Dalal, S., Seth, B., Radulescu, M., Secara, C., & Tolea, C. (2022). Predicting Fraud in Financial Payment Services through Optimized Hyper-Parameter-Tuned XGBoost Model. *Mathematics*, *10*(24). https://doi.org/10.3390/math10244679

Deshpande, Anand., & Kumar, Manish. (2018). *Artificial intelligence for big data : complete guide to automating big data solutions using artificial intelligence techniques.* Packt Publishing.

Erjavac, I., Kalafatovic, D., & Mauša, G. (2022). Coupled encoding methods for antimicrobial peptide prediction: How sensitive is a highly accurate model? *Artificial Intelligence in the Life Sciences*, *2*, 100034. https://doi.org/10.1016/j.ailsci.2022.100034

Hasan, M. K., Jawad, M. T., Dutta, A., Awal, M. A., Islam, M. A., Masud, M., & Al-Amri, J. F. (2021). Associating Measles Vaccine Uptake Classification and its Underlying Factors Using an Ensemble of Machine Learning Models. *IEEE Access*, *9*, 119613–119628. https://doi.org/10.1109/ACCESS.2021.3108551

Jolly, K. (2018). *Machine Learning with scikit-learn quick start guide : classification, regression, and clustering techniques in Python* (J. Topiwala, A. Varangaonkar, A. Gour, S. Carvallo, Safis, T. Daruwale Soni, & J. Monteiro, Eds.; 1st ed., Vol. 1). Packt Publishing Ltd.

Kaushik, S., & Birok, R. (2021). Heart Failure prediction using Xgboost algorithm and feature selection using feature permutation. *2021 4th International Conference on Electrical, Computer and Communication Technologies, ICECCT 2021*. https://doi.org/10.1109/ICECCT52121.2021.9616626

Kavzoglu, T., & Teke, A. (2022). Advanced hyperparameter optimization for improved spatial prediction of shallow landslides using extreme gradient boosting (XGBoost). *Bulletin of Engineering Geology and the Environment*, *81*(5). https://doi.org/10.1007/s10064-022-02708-w

Kurikulum developer dicoding. (2023, April 19). *Machine Learning Terapan: Model Development dengan Boosting Algorithm*. Dicoding. https://www.dicoding.com/academies/319/tutorials/18590

Li, S., & Zhang, X. (2020). Research on orthopedic auxiliary classification and prediction model based on XGBoost algorithm. *Neural Computing and Applications*, *32*(7), 1971–1979. https://doi.org/10.1007/s00521-019-04378-4

Li, X., Shan, G., & Shek, C. H. (2022). Machine learning prediction of magnetic properties of Fe-based metallic glasses considering glass forming ability. *Journal of Materials Science and Technology*, *103*, 113–120. https://doi.org/10.1016/j.jmst.2021.05.076

Ma, Z., Chang, H., Sun, Z., Liu, F., Li, W., Zhao, D., & Chen, C. (2020). Very Short-Term Renewable Energy Power Prediction Using XGBoost Optimized by TPE Algorithm. *2020 4th International Conference on HVDC, HVDC 2020*, 1236–1241. https://doi.org/10.1109/HVDC50696.2020.9292870

Navas, J. (2022, February 8). *What is hyperparameter tuning\_ \_ Anyscale*. Anyscale. https://www.anyscale.com/blog/what-is-hyperparameter-tuning

Nguyen, H., Vu, T., Vo, T. P., & Thai, H. T. (2021). Efficient machine learning models for prediction of concrete strengths. *Construction and Building Materials*, *266*. https://doi.org/10.1016/j.conbuildmat.2020.120950

Nyoman, N., Pinata, P., Sukarsa, M., Kadek, N., & Rusjayanthi, D. (2020). Prediksi Kecelakaan Lalu Lintas di Bali dengan XGBoost pada Python. *Jurnal Ilmiah Merpati*, *8*(3), 188–196.

Pham, K., Kim, D., Le, C. V., & Choi, H. (2022). Dual tree-boosting framework for estimating warning levels of rainfall-induced landslides. *Landslides*, *19*(9), 2249–2262. https://doi.org/10.1007/s10346-022-01894-8

Qin, C., Zhang, Y., Bao, F., Zhang, C., Liu, P., & Liu, P. (2021). XGBoost optimized by adaptive particle swarm optimization for credit scoring. *Mathematical Problems in Engineering*, *2021*. https://doi.org/10.1155/2021/6655510

scikit learn Developers. (2023). *Standar Scaler scikit-learn*. Scikit Learn Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Sunata, H., Azrullah, F. J., & Rianto, Y. (2020). Komparasi Tujuh Algoritma Identifikasi Fraud ATM Pada PT. Bank Central Asia Tbk. *Jurnal Teknik Inforamatika Dan Sistem Informasi*, *7*(3), 2407–4322. http://jurnal.mdp.ac.id

Tankari, M. R. (2020). Rainfall variability and farm households’ food insecurity in Burkina Faso: nonfarm activities as a coping strategy. *Food Security*, *12*, 567–578. https://doi.org/10.1007/s12571-019-01002-0/Published

Tattar, P. (2018). *Hands-on ensemble learning with R : a beginner’s guide to combining the power of machine learning algorithms using ensemble techniques* (S. Shetty, T. Gupta, A. Singh, D. Chaudhary, Safis, M. Chettiyer, & J. Chirayil, Eds.; 1st ed., Vol. 1). Packt Publishing Ltd.

Xiang, Y., Gou, L., He, L., Xia, S., & Wang, W. (2018). A SVR–ANN combined model based on ensemble EMD for rainfall prediction. *Applied Soft Computing Journal*, *73*, 874–883. https://doi.org/10.1016/j.asoc.2018.09.018

Yu, Y., Zhu, J., Gao, T., Liu, L., Yu, F., Zhang, J., & Wei, X. (2022). Evaluating the influential variables on rainfall interception at different rainfall amount levels in temperate forests. *Journal of Hydrology*, *615*. https://doi.org/10.1016/j.jhydrol.2022.128572

Zhang, D., & Gong, Y. (2020). The Comparison of LightGBM and XGBoost Coupling Factor Analysis and Prediagnosis of Acute Liver Failure. *IEEE Access*. https://doi.org/10.1109/ACCESS.2020.3042848

Lampiran 1. Pengkodean klasifikasi *binary* dan *multiclass*

## Data preprocessing

"""

import pandas as pd

import seaborn as sns

import numpy as np

"""### Data Loading"""

data\_belum\_clean = pd.read\_excel("../2022 Feb 2023 perjam.xlsx")

data\_belum\_clean

data\_belum\_clean.describe()

data\_classified = pd.read\_excel("../data/data\_classified.xlsx")

data\_classified

data\_belum\_clean.info()

data\_baru = pd.read\_excel("../Data Des 2022 to Feb 2023 perjam Clean.xlsx")

data\_baru = data\_baru.drop("klasifikasi", axis=1)

data\_baru

data\_baru.info()

# data\_baru.sort\_values(by=['Date and Time'], inplace=True, ascending=True)

data\_baru.set\_index("Date and Time", inplace=True)

"""#### Missing value"""

data\_baru.describe()

conditions = [

    (data\_baru["Precip 1Hr (mm) M"] == 0),

    (data\_baru["Precip 1Hr (mm) M"] > 0),

]

values = ["tidak hujan", "hujan"]

data\_baru["classification"] = np.select(conditions, values)

data\_baru

data\_baru.isnull().sum()

"""#### Pesebaran data

dilihat dengan histogram

"""

import matplotlib.pyplot as plt

data\_baru.hist(bins=50, figsize=(20, 15))

plt.show()

"""#### Matriks korelasi"""

plt.figure(figsize=(5, 5))

correlation\_matrix = data\_baru.corr().round(2)

sns.heatmap(

    data=correlation\_matrix,

    annot=True,

    cmap="coolwarm",

    linewidths=0.5,

)

# Get the correlation coefficients for "rain\_sum" feature

rain\_corr = data\_baru.corr()["Precip 1Hr (mm) M"].sort\_values(ascending=False)

rain\_corr

"""### Data preparation

#### one-hot encoding

mengubah data kategori dengan one-hot encoding

"""

data\_baru = pd.get\_dummies(data\_baru)

data\_baru

data\_baru.info()

"""Membagi data train dataset"""

data\_baru.info()

"""pertama untuk data fitur hapus kolom precip, dan klasifikasi"""

from sklearn import preprocessing

df\_new = data\_baru

X = df\_new.drop(df\_new.columns[[1, 5, 6]], axis=1)

y = df\_new[["classification\_hujan", "classification\_tidak hujan"]]

from sklearn.model\_selection import train\_test\_split

##Data dibagi menjadi 4 case train-test yaitu 60-40, 70-30, 80-20, dan 90-10

##tinggal mengganti nilai train dan test nya

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, train\_size=0.60, test\_size=0.40, random\_state=50, shuffle=True

)

print("train\_hujan", (y\_train["classification\_hujan"] == 1).sum())

print("train\_tidak hujan", (y\_train["classification\_tidak hujan"] == 1).sum())

print("\ntest\_hujan", (y\_test["classification\_hujan"] == 1).sum())

print("test\_tidak hujan", (y\_test["classification\_tidak hujan"] == 1).sum())

print("\ntotal hujan", (y["classification\_hujan"] == 1).sum())

print("total tidak hujan", (y["classification\_tidak hujan"] == 1).sum())

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

"""Jumlah data train dan test"""

print(X\_train.shape)

print(X\_test.shape)

"""## Model Development"""

from sklearn.model\_selection import RandomizedSearchCV, GridSearchCV

from xgboost import XGBClassifier

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import (

    accuracy\_score,

    precision\_score,

    recall\_score,

    f1\_score,

    roc\_auc\_score,

    auc,

    roc\_curve,

)

import plotly.express as px

import plotly.graph\_objs as go

"""##### Tampa tuning"""

xgb = XGBClassifier(

    max\_depth=6,

    n\_estimators=100,

    learning\_rate=0.3,

)

xgb.fit(X\_train, y\_train)

"""##### Dengan Tuning

mendefenisikan hyperparameter

"""

from scipy.stats import uniform, randint

import numpy as np

params = {"n\_estimators": [100], "max\_depth": [6], "learning\_rate": uniform(0.01, 1)}

params

"""Tuning dengan RSCV"""

# coarse search

# Define the XGBoost model

xgb\_model = XGBClassifier()

# Perform random search

search = RandomizedSearchCV(

    xgb\_model,

    param\_distributions=params,

    n\_iter=50,

    cv=10,

    scoring="accuracy",

    n\_jobs=-1,

)

search.fit(X\_train, y\_train)

print("Best hyperparameters: ", search.best\_params\_)

print("Accuracy", search.best\_score\_)

# fine search

best\_params = search.best\_params\_

param\_dist = {

    "n\_estimators": [100],

    "max\_depth": [6],

    "learning\_rate": np.logspace(

        np.log10(best\_params["learning\_rate"] - 0.01),

        np.log10(best\_params["learning\_rate"] + 0.1),

        50,

    ),

}

fine\_search = GridSearchCV(

    xgb\_model,

    param\_grid=param\_dist,

    cv=10,

    scoring="accuracy",

    n\_jobs=-1,

)

fine\_search.fit(X\_train, y\_train)

print("Best hyperparameters: ", fine\_search.best\_params\_)

print("Accuracy", fine\_search.best\_score\_)

"""Hasil tuning

### Model Evaluation

"""

X\_test = scaler.transform(X\_test)

"""##### Tampa Tuning"""

pred\_no\_tuning = xgb.predict(X\_test)

cm = confusion\_matrix(y\_test.values.argmax(axis=1), pred\_no\_tuning.argmax(axis=1))

print(cm)

sns.heatmap(cm, annot=True)

# calculate the accuracy, precision, recall, and F1-score of the model

accuracy = accuracy\_score(y\_test, pred\_no\_tuning)

precision = precision\_score(y\_test, pred\_no\_tuning, average="weighted")

recall = recall\_score(y\_test, pred\_no\_tuning, average="weighted")

f1 = f1\_score(y\_test, pred\_no\_tuning, average="weighted")

# print the evaluation metrics

print(f"Accuracy: {accuracy:.2f}")

print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

print(f"F1-score: {f1:.2f}")

print(classification\_report(y\_test, pred\_no\_tuning))

"""##### Dengan tuning"""

y\_pred = fine\_search.best\_estimator\_.predict(X\_test)

cm = confusion\_matrix(y\_test.values.argmax(axis=1), y\_pred.argmax(axis=1))

print(cm)

sns.heatmap(cm, annot=True)

# calculate the accuracy, precision, recall, and F1-score of the model

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average="weighted")

recall = recall\_score(y\_test, y\_pred, average="weighted")

f1 = f1\_score(y\_test, y\_pred, average="weighted")

# print the evaluation metrics

print(f"Accuracy: {accuracy:.2f}")

print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

print(f"F1-score: {f1:.2f}")

print(classification\_report(y\_test, y\_pred))

data\_baru.info()

##check jumlah data encoding masing masing kategory

conditions = [

    (data\_baru["classification\_hujan"] == 1),

    (data\_baru["classification\_tidak hujan"] == 1),

]

values = ["hujan", "tidak hujan"]

data\_baru["classification"] = np.select(conditions, values)

data\_baru.head()

"""#### Visualisasi Tuning"""

result\_coarse = search.cv\_results\_

result\_fine = fine\_search.cv\_results\_

params\_key = [f"param\_{p}" for p in params.keys()]

params\_key.append("Accuracy")

params\_key

Accuracy = []

for mean\_score in result\_coarse["mean\_test\_score"]:

    Accuracy.append(mean\_score)

accuracy = {}

accuracy["Accuracy"] = Accuracy

result\_coarse["Accuracy"] = Accuracy

Accuracy\_Fine = []

for mean\_score in result\_fine["mean\_test\_score"]:

    Accuracy\_Fine.append(mean\_score)

accuracy\_fine = {}

accuracy\_fine["Accuracy"] = Accuracy\_Fine

result\_fine["Accuracy"] = Accuracy\_Fine

import matplotlib.pyplot as plt

plt.figure(figsize=(6, 6))

x1 = result\_coarse["param\_learning\_rate"]

y1 = result\_coarse["mean\_test\_score"]

x2 = result\_fine["param\_learning\_rate"]

y2 = result\_fine["mean\_test\_score"]

plt.scatter(x1, y1, color="red", label="Random search")

plt.scatter(x2, y2, color="blue", label="Grid search")

plt.legend()

plt.xlabel("Learning rate")

plt.ylabel("Accuracy")

# Tampilkan grafik

plt.show()

## melakukan prediksi 24 jam kedepan

last\_24\_jam = X.iloc[-24:]

fine\_search.best\_estimator\_.predict(last\_24\_jam)

Lampiran 2. Pengkodean *multiclass classification*

import pandas as pd

import seaborn as sns

import numpy as np

"""### Data Loading"""

data\_baru = pd.read\_excel("../Data Des 2022 to Feb 2023 perjam Clean.xlsx")

data\_baru = data\_baru.drop("klasifikasi", axis=1)

data\_baru.head()

data\_baru.info()

data\_baru.set\_index("Date and Time", inplace=True)

"""#### Missing value"""

data\_baru.describe()

"""\_ada lebih dari 100 data yang hilang, total data adalah 2039, dan yang terhitung 1900\_"""

conditions = [

    (data\_baru["Precip 1Hr (mm) M"] == 0),

    (data\_baru["Precip 1Hr (mm) M"] > 0) & (data\_baru["Precip 1Hr (mm) M"] <= 5),

    (data\_baru["Precip 1Hr (mm) M"] >= 5) & (data\_baru["Precip 1Hr (mm) M"] <= 50),

    (data\_baru["Precip 1Hr (mm) M"] >= 50) & (data\_baru["Precip 1Hr (mm) M"] <= 100),

    (data\_baru["Precip 1Hr (mm) M"] >= 100),

]

values = ["tidak hujan", "sangat ringan", "ringan dan sedang", "lebat", "sangat lebat"]

data\_baru["classification"] = np.select(conditions, values)

data\_baru.head()

# Pengecekan missing value

data\_baru.isnull().sum()

"""#### Pesebaran data

dilihat dengan histogram

"""

import matplotlib.pyplot as plt

data\_baru.hist(bins=50, figsize=(7, 20), layout=(5, 1))

plt.show()

"""#### Matriks korelasi"""

plt.figure(figsize=(5, 5))

correlation\_matrix = data\_baru.corr().round(2)

sns.heatmap(

    data=correlation\_matrix,

    annot=True,

    cmap="coolwarm",

    linewidths=0.5,

)

# Get the correlation coefficients for "rain\_sum" feature

rain\_corr = data\_baru.corr()["Precip 1Hr (mm) M"].sort\_values(ascending=False)

rain\_corr

"""### Data preparation

#### one-hot encoding

mengubah data kategori dengan one-hot encoding

"""

data\_baru = pd.get\_dummies(data\_baru)

data\_baru

data\_baru.info()

"""Membagi data train dataset"""

data\_baru.info()

"""pertama untuk data fitur hapus kolom precip, dan klasifikasi"""

from sklearn import preprocessing

df\_new = data\_baru

X = df\_new.drop(df\_new.columns[[1, 5, 6, 7]], axis=1)

y = df\_new[

    [

        "classification\_ringan dan sedang",

        "classification\_sangat ringan",

        "classification\_tidak hujan",

    ]

]

from sklearn.model\_selection import train\_test\_split

##Data dibagi menjadi 4 case train-test yaitu 60-40, 70-30, 80-20, dan 90-10

##tinggal mengganti nilai train dan test nya

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, train\_size=0.60, test\_size=0.40, random\_state=50, shuffle=True

)

print("train\_ringan sedang", (y\_train["classification\_ringan dan sedang"] == 1).sum())

print("train\_sangat ringan", (y\_train["classification\_sangat ringan"] == 1).sum())

print("train\_tidak hujan", (y\_train["classification\_tidak hujan"] == 1).sum())

print(

    "\ntest\_ringan dan sedang", (y\_test["classification\_ringan dan sedang"] == 1).sum()

)

print("test\_sangat ringan", (y\_test["classification\_sangat ringan"] == 1).sum())

print("test\_tidak hujan", (y\_test["classification\_tidak hujan"] == 1).sum())

print("\ntotal ringan dan sedang", (y["classification\_ringan dan sedang"] == 1).sum())

print("total sangat ringan", (y["classification\_sangat ringan"] == 1).sum())

print("total tidak hujan", (y["classification\_tidak hujan"] == 1).sum())

"""##### Standarisasi data train"""

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

pd.DataFrame(X\_train).describe().round(2)

"""Jumlah data train dan test"""

print(X\_train.shape)

print(X\_test.shape)

"""## Model Development"""

from sklearn.model\_selection import RandomizedSearchCV, GridSearchCV

from xgboost import XGBClassifier

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import (

    accuracy\_score,

    precision\_score,

    recall\_score,

    f1\_score,

    roc\_auc\_score,

    auc,

    roc\_curve,

)

import plotly.express as px

import plotly.graph\_objs as go

"""##### Tampa tuning"""

xgb = XGBClassifier(

    max\_depth=6,

    n\_estimators=100,

    learning\_rate=0.3,

)

xgb.fit(X\_train, y\_train)

"""##### Dengan Tuning

mendefenisikan hyperparameter

"""

from scipy.stats import uniform, randint

import numpy as np

params = {"n\_estimators": [100], "max\_depth": [6], "learning\_rate": uniform(0.01, 1)}

params

"""##### Tuning dengan RSCV"""

# Define the XGBoost model

xgb\_model = XGBClassifier()

# Perform random search

search = RandomizedSearchCV(

    xgb\_model,

    param\_distributions=params,

    n\_iter=50,

    cv=10,

    scoring="accuracy",

    n\_jobs=-1,

)

search.fit(X\_train, y\_train)

print("Best hyperparameters: ", search.best\_params\_)

print("Accuracy", search.best\_score\_)

# fine search

best\_params = search.best\_params\_

param\_dist = {

    "n\_estimators": [100],

    "max\_depth": [6],

    "learning\_rate": np.logspace(

        np.log10(best\_params["learning\_rate"] - 0.01),

        np.log10(best\_params["learning\_rate"] + 0.1),

        50,

    ),

}

fine\_search = GridSearchCV(

    xgb\_model,

    param\_grid=param\_dist,

    cv=10,

    scoring="accuracy",

    n\_jobs=-1,

)

fine\_search.fit(X\_train, y\_train)

print("Best hyperparameters: ", fine\_search.best\_params\_)

print("Accuracy", fine\_search.best\_score\_)

"""### Model Evaluation

#### Standarisasi data test

"""

X\_test = scaler.transform(X\_test)

pd.DataFrame(X\_test).describe().round(2)

"""#### tampa tuning"""

pred\_no\_tuning = xgb.predict(X\_test)

cm = confusion\_matrix(y\_test.values.argmax(axis=1), pred\_no\_tuning.argmax(axis=1))

print(cm)

sns.heatmap(cm, annot=True)

# calculate the accuracy, precision, recall, and F1-score of the model

accuracy = accuracy\_score(y\_test, pred\_no\_tuning)

precision = precision\_score(y\_test, pred\_no\_tuning, average="weighted")

recall = recall\_score(y\_test, pred\_no\_tuning, average="weighted")

f1 = f1\_score(y\_test, pred\_no\_tuning, average="weighted")

# print the evaluation metrics

print(f"Accuracy: {accuracy:.2f}")

print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

print(f"F1-score: {f1:.2f}")

print(classification\_report(y\_test, pred\_no\_tuning))

"""#### Dengan tuning"""

y\_pred = fine\_search.best\_estimator\_.predict(X\_test)

cm = confusion\_matrix(y\_test.values.argmax(axis=1), y\_pred.argmax(axis=1))

print(cm)

sns.heatmap(cm, annot=True)

# calculate the accuracy, precision, recall, and F1-score of the model

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average="weighted")

recall = recall\_score(y\_test, y\_pred, average="weighted")

f1 = f1\_score(y\_test, y\_pred, average="weighted")

# print the evaluation metrics

print(f"Accuracy: {accuracy:.2f}")

print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

print(f"F1-score: {f1:.2f}")

print(classification\_report(y\_test, y\_pred))

##check jumlah data encoding masing masing kategory

conditions = [

    (data\_baru["classification\_ringan dan sedang"] == 1),

    (data\_baru["classification\_sangat ringan"] == 1),

    (data\_baru["classification\_tidak hujan"] == 1),

]

values = ["ringan dan sedang", "sangat ringan", "tidak hujan"]

data\_baru["classification"] = np.select(conditions, values)

data\_baru.head()

"""#### Visualisasi Tuning"""

result\_coarse = search.cv\_results\_

result\_fine = fine\_search.cv\_results\_

params\_key = [f"param\_{p}" for p in params.keys()]

params\_key.append("Accuracy")

params\_key

Accuracy = []

for mean\_score in result\_coarse["mean\_test\_score"]:

    Accuracy.append(mean\_score)

accuracy = {}

accuracy["Accuracy"] = Accuracy

result\_coarse["Accuracy"] = Accuracy

Accuracy\_Fine = []

for mean\_score in result\_fine["mean\_test\_score"]:

    Accuracy\_Fine.append(mean\_score)

accuracy\_fine = {}

accuracy\_fine["Accuracy"] = Accuracy\_Fine

result\_fine["Accuracy"] = Accuracy\_Fine

import matplotlib.pyplot as plt

plt.figure(figsize=(6, 6))

x1 = result\_coarse["param\_learning\_rate"]

y1 = result\_coarse["mean\_test\_score"]

x2 = result\_fine["param\_learning\_rate"]

y2 = result\_fine["mean\_test\_score"]

plt.scatter(x1, y1, color="red", label="Random search")

plt.scatter(x2, y2, color="blue", label="Grid search")

plt.legend()

plt.xlabel("Learning rate")

plt.ylabel("Accuracy")

# Tampilkan grafik

plt.show()

##prediksi 24 jam kedepan

last\_24\_jam = X.iloc[-24:]

fine\_search.best\_estimator\_.predict(last\_24\_jam)