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

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	: Dr.rer.nat. Martarizal,	(……)
Penguji II	: Drs. Sastra Kusuma Wijaya, Ph.D.	(……)

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, sangat 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 sering kali 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) Bapak Dr. Djati Handoko dan Bapak Maulana Putra, S.Si., M.T., selaku pembimbing selama berjalannya tugas akhir yang telah memberikan penerangan bagi penulis dalam penyelesaian tugas akhir ini.
3) Bapak Drs. Sastra Kusuma Wijaya, Ph.D. dan Bapak 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), serta kucing-kucing peliharaan kos (Zi long, Marjan, Manya dan lainnya) yang selalu mendukung dalam bentuk moril maupun materil.
8) Bang Noval, Bang Amaik, Bang Rahmat, Bang Agum, Bang Ryan, Ridho, Rahman, Popo, Syakur, Ababil 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 staf 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 (Bang Egi, Bang Ai, Bang Syafei, 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, mengalih media/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.

Klasifikasi curah hujan sangat membantu masyarakat dan instansi terkait dalam mengambil kebijakan seperti pengelolaan 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 data set 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 for rainfall 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 class*ification case, which is equal to 0.047 with an accuracy of 90.25% of the *train*ing data.

Key words:

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

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 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 menganalisis 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, di antaranya *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 *eXtreme Gradient Boosting* (*XGBoost*). Dalam memahami data, *XGBoost* memiliki keunggulan dalam melakukan observasi terhadap data yang memiliki statistik tidak linear (Shahani et al., 2021), sehingga model mampu melakukan prediksi regresi maupun klasifikasi dengan baik. Selain pemahaman data yang bagus, kelebihan lainnya yang dimiliki *XGBoost* adalah dapat mengatasi *missing value*, *outliers*, dan nilai kardinalitas yang tinggi pada data kategori (Bansal et al., 2023).

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 di luar 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 di sini *hyperparameter* akan mencari kombinasi secara tepat (Navas, 2022).

1.1 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?

1.2 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 *eXxtreme Gradient Boosting*.

1.3 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.

1.4 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.

1.5 Sistematika Penulisan

Sistematika penulisan naskah penelitian terdiri dari lima bab, di antaranya:

BAB I PENDAHULUAN

Membahas latar belakang, 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 penelitian selanjutnya.

BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 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 eksplisit.

Secara garis besar, *machine learning* dapat dikategorikan menjadi tiga jenis utama (Das & Mert Cakmak, 2018):

1. *Supervised Learning*: yaitu bentuk pembelajaran mesin di mana data 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.
2. *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, data tidak memiliki label seperti pada *supervised learning*. Dengan demikian, algoritma biasanya akan menggunakan metrik seperti jarak untuk mengelompokkan data bersama-sama, bergantung pada seberapa jauh mereka satu sama lain.
3. *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 model 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.

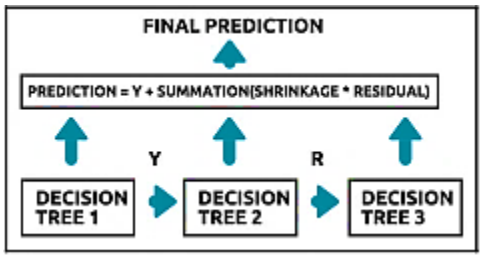
2.1.2 Gradient Boosting

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:



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

Inilah yang terjadi pada diagram tersebut (Jolly, 2018):

1. *Decision tree* pertama dilatih dengan data set yang telah dikumpulkan, 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 ditetapkan.
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 digunakan untuk mengontrol laju proses *Gradient boosting* yang diinginkan.
6. Nilai penyusutan (*learning_rate*) yang kecil menunjukkan bahwa algoritma akan belajar lebih cepat, dan oleh karena itu, harus di kompensasi 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.

2.1.3 eXtreme Gradient Boosting (XGBoost)

Pada tahun 2014, Dr. Tianqi Chen yang berasal dari Universitas Washington mengembangkan sebuah model bernama *XGBoost* yang menjadikan *gradient boosting* sebagai dasar dalam membangun teorinya (Zhang & Gong, 2020). *Gradient boosting* merupakan algoritma yang dapat mengurangi fungsi *loss* dan menjadi 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 kategori 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.

$$obj(\vartheta) = L(\vartheta) + \Omega(\vartheta) \tag{2.1}$$

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

$$L(\theta) = \sum_{l=1}^{n} l(y_i, p_i) \tag{2.2}$$

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

Diasumsikan data adalah $D = \{(x_i, y_i)\}(|D| = n, x_i \in R^m, y_i \in R)$, jadi ada n observasi. Setiap observasi memiliki m fitur dan variabel y. Maka nilai prediksi yang akan diperoleh model adalah $\hat{y}_i$, adalah seperti persamaan 2.3:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{k} f_k(x_i) \tag{2.3}$$

Di sini f_k adalah *regression tree* dan $f_k(x_1)$ adalah skor yang disediakan *k-tree* untuk observasi ke-i. Fungsi f_k dipilih untuk meminimumkan nilai fungsi objektif seperti persamaan 2.4:

$$L(\phi) = \sum_{l} l(y_{i,}\, p_{i,}) + \sum_{k} \Omega(f_k) \tag{2.4}$$

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

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \left|\left|\mathrm{w}\right|\right|^2 \tag{2.5}$$

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

$$L^t = \sum_{l=1}^{n} l\left(y_i, {p_i}^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{2.6}$$

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

$$L_{split} = \left[\frac{\left(\sum_{n\in I_R} g_n\right)^2}{\sum_{n\in I_R} h_n + \lambda}\right] + \frac{1}{2}\left[\frac{\left(\sum_{n\in I_L} g_n\right)^2}{\sum_{n\in I_L} h_n + \lambda}\right] + \left[\frac{(\sum_{n\in I} g_n)^2}{\sum_{n\in I} h_n + \lambda}\right] - \gamma \tag{2.7}$$

Di mana *l* mewakili *subset* dari pengamatan yang tersedia di *node* saat ini. I_L adalah hipunan bagian dari observasi yang tersedia di *node* kiri dan I_R merupakan hipunan bagian dari observasi yang tersedia di *node* kanan setelah *split*. Fungsi g_n dan h_n digunakan untuk menemukan *split* terbaik, dan didefinisikan sebagai persamaan 2.8 dan 2.9:

$$g_n = \partial_{{p_n}^{(j-1)}}\, l\left(y_n,\, {p_n}^{(j-1)}\right) \tag{2.8}$$

$$h_n = \partial^2{}_{{p_n}^{(j-1)}}\, l\left(y_n,\, {p_n}^{(j-1)}\right) \tag{2.9}$$

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*, maka parameter yang akan digunakan adalah *num_boost_rounds* daripada *n_estimators*.

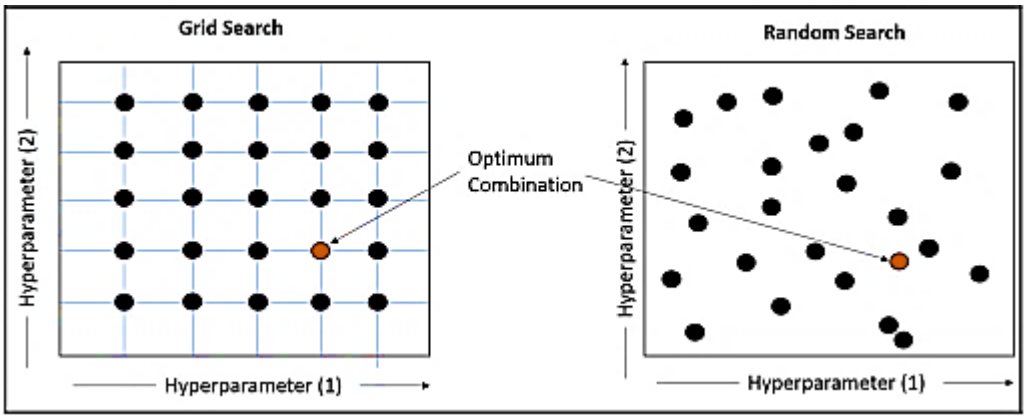
2.1.4 Hyperparameter Tuning

Model *machine learning* 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 sebenarnya. *Grid Search* merupakan metode untuk penyesuaian *hyperparameter* dengan cara "*brute force*". Dalam metode ini, pencarian dilakukan dengan 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

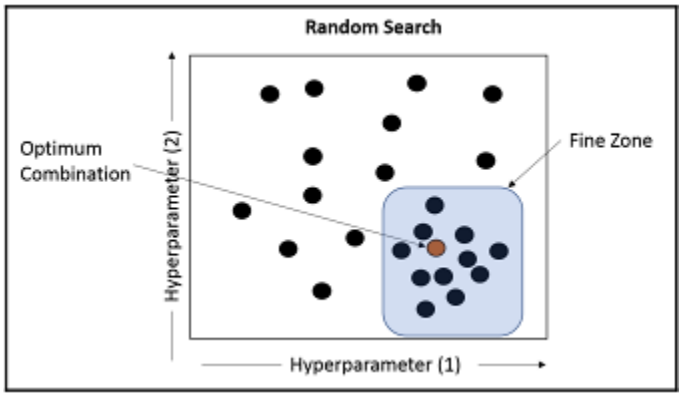
dalam kecepatannya. Menyesuaikan model dengan setiap kombinasi membutuhkan kapasitas komputasi yang tinggi dan waktu yang signifikan, sehingga proses akan sangat lama dan tidak efisien terutama jika sumber daya komputasi terbatas (Navas, 2022).

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.



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.



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.

2.2 Penelitian Terkait

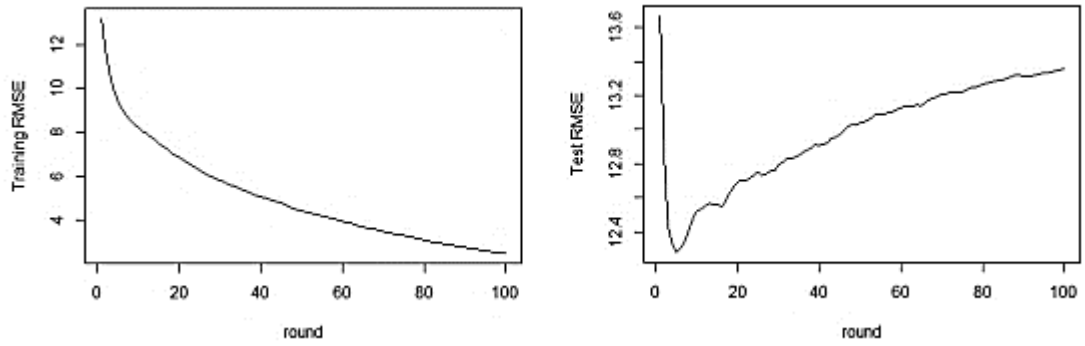
2.2.1 Rainfall Prediction Using eXtreme Gradient Boosting.

Penelitian mengenai prediksi curah hujan menggunakan *eXtreme Gradient Boosting (XGBoost)* 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 data set, akan tetapi yang digunakan hanya 8 fitur saja seperti disajikan dalam tabel 2.1. Data set *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 *train*ing. Hal ini ditunjukkan pada gambar 2.4. Hal ini menunjukkan kecenderungan *overvitting XGBoost* seiring bertambahnya iterasi.



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

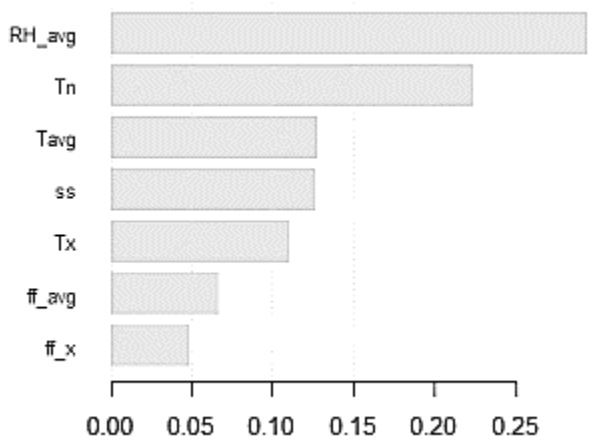
Tabel 2.2 menunjukkan bahwa nilai kesalahan pelatihan, *10-fold cross validation* menghasilkan kesalahan yang lebih rendah daripada menggunakan data set *full training* yang menunjukkan indikasi *overfitting*. Ketika data non-hujan dikecualikan kesalahannya meningkat. Kesalahan yang lebih tinggi ini 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

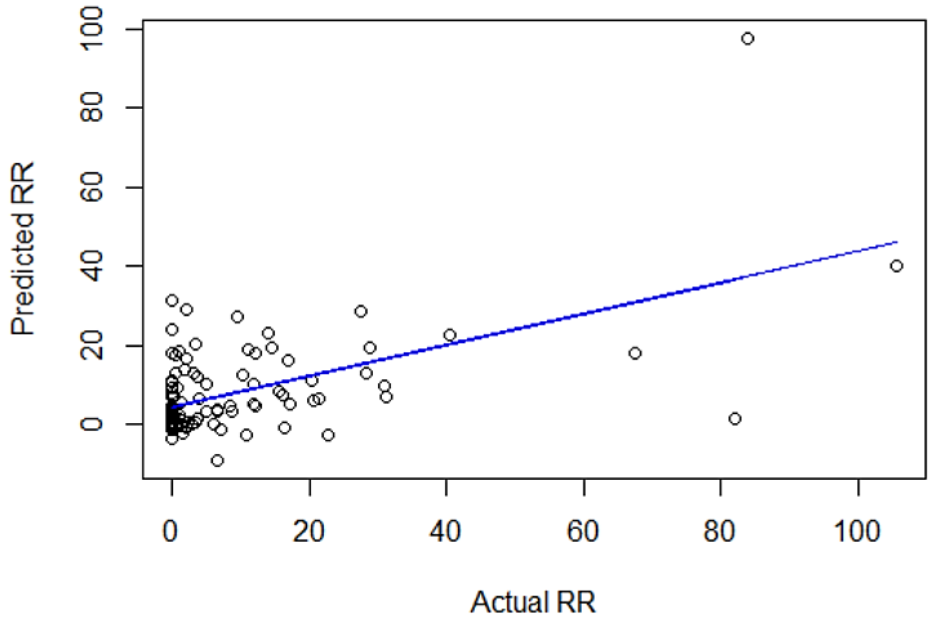
XGBoost mampu memeringkat atribut penting yang berkontribusi pada model. Pemeringkatan atribut ditunjukkan pada Gambar 2.5 Ini menunjukkan bahwa model prediksi curah hujan sangat dipengaruhi oleh kelembaban relatif rata-rata (*RH_avg*) dan suhu minimum (Tn). Hasil ini sejalan dengan penelitian sebelumnya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



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 data set 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 pengindraan jauh untuk membuat model yang lebih akurat.



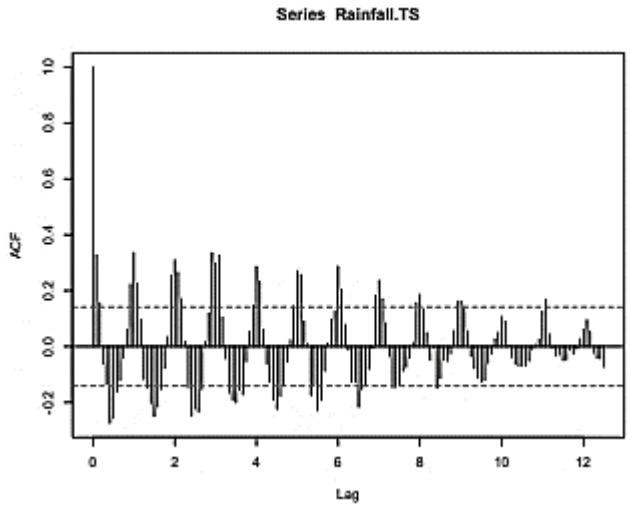
Gambar 2. 6 RR prediksi versus RR sebenarnya (Anwar et al., 2021).

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 menunjukkan 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.

2.2.2 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 menentukan 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.



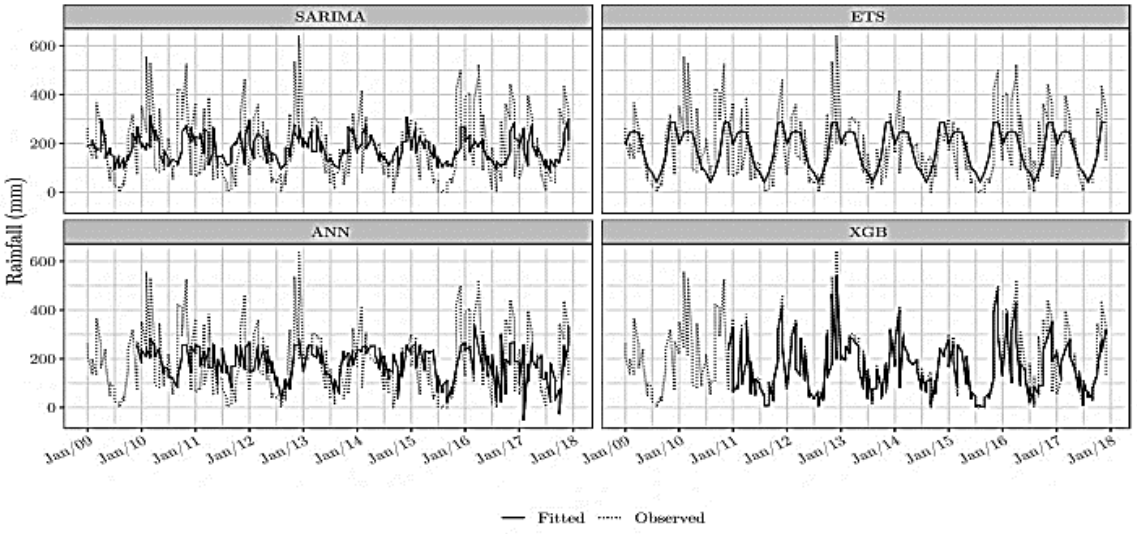
Gambar 2. 7 *Auto Correlation Function plot* untuk data *train*ing (Agata & Jaya, 2019).

Gambar di atas menunjukkan *Autocorrelation function* dengan pola musim yang jelas dengan *time lag* 12 (s = 12 bulan). Ini menunjukkan 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).

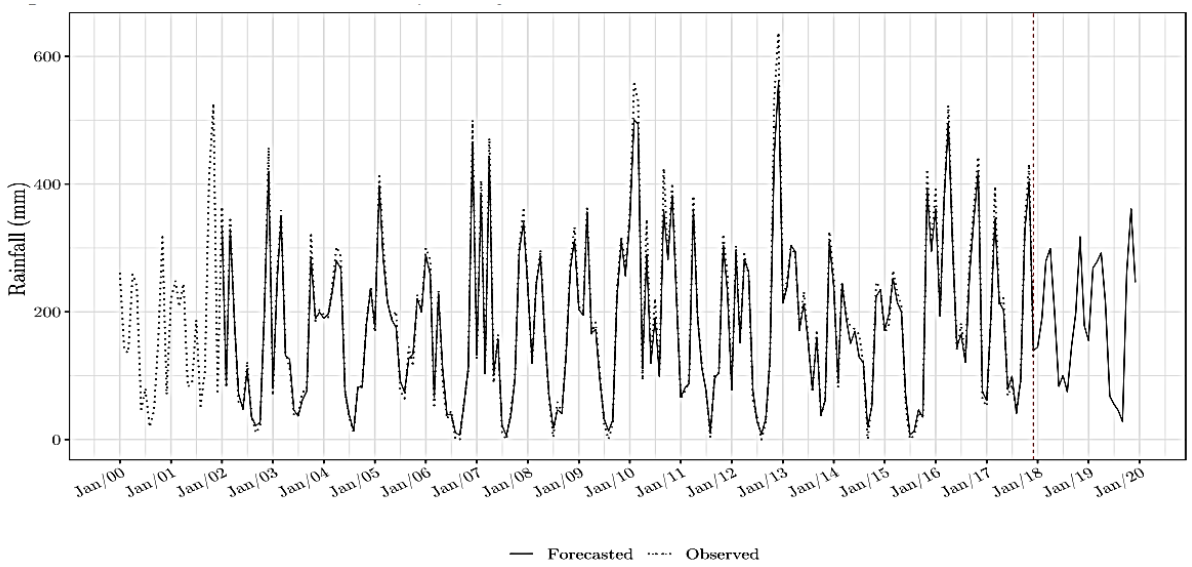
Model	MAD	RMSED	MAPE
SARIMA	116.9017	162.6560	65.7756
exponential smoothing	117.1898	158.5143	64.8473
Artificial Neural Network	117.1898	158.5143	64.8473
XGBoost	112.5225	152.8657	63.7136

Berdasarkan tabel di atas dapat diketahui 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.



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

Secara jelasnya bisa dilihat pada gambar 2.9.



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

2.2.3 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.

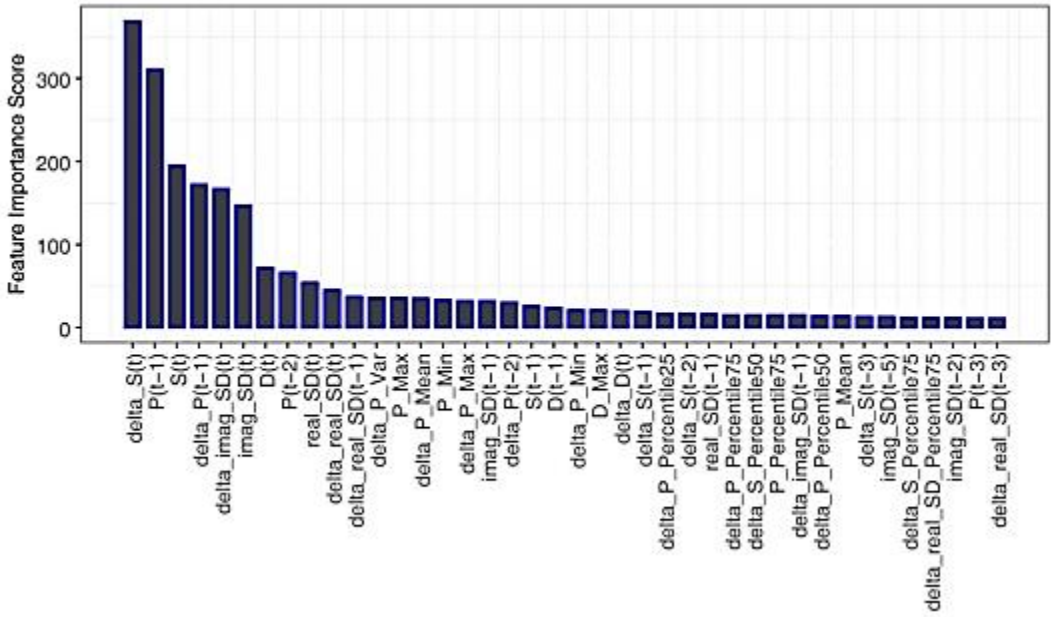
Data kecepatan angin dan arah angin dihasilkan pada menara anemometer. Tenaga angin yang dihasilkan diukur melalui sistem SCADA. Semua data yang digunakan dalam penelitian ini dirata-ratakan dan disimpan dengan interval 10 menit, mulai dari '1 Januari

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 Data set 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.



Gambar 2. 10 Skor kepentingan fitur data set (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

Berdasarkan data tabel di atas dapat dilihat bahwa model *XGBoost* memiliki akurasi prediksi yang tinggi dengan nilai NRMSE dan MAPE terendah.

2.2.4 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 (XGBoost)* 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 beberapa 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 *XGBoost* dapat ditemukan nilai optimalnya dalam ruang pencarian kontinu. Namun, *Particle swarm optimization* klasik cenderung jatuh ke optimal lokal. Untuk mengatasi masalah ini, Chao Qin et al. mengusulkan model penilaian kredit *XGBoost* 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, algoritma 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 (*Tree-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* (eror tipe I dan eror tipe II), *Brier score*, dan 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 parameter 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* Parameter lainnya sama dengan *Decision Tree*
Adaptive Boosting-Neural Network, Bagging-Neural Network	*N estimator = 100* Sisanya 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 data set bisa dilihat pada tabel 2.8.

Table 2. 8 Tabel hasil performa model untuk data set 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

2.2.5 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 kekukuhan 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 *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

2.2.6 Mengaitkan 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*. Data set 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.

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)	0.582 ± 0.018
BNB	Parameter halus *laplace* (*alpha* = 0.0), fitur *binarizing threshold* (*binarize* = 0.4), dan kelas *prior* (*fit_prior* = *true*)	0.582 ± 0.022
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)	0.725 ± 0.027
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)	0.616 ± 0.029
XGBoost	Jumlah *hessian* (*min_child_weight* = 1.0), *loss* minimum (*gamma* = 1.0), *rasion instance subsample* (*subsample* = 0.25), rasio *subsample* kolom (*colsample_bytree* = 1.0), kedalaman pohon (*max_depth* = 5)	0.769 ± 0.014
LGBoost	*Base learner* daun pohon (*num_leaves* = 25), jumlah pohon (*n_estimators* = 50), rasio *subsample* (*subsample* = 0.25), rasio	0.778 ± 0.020

2.2.7 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 memungkinkan 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 data set kekuatan tekan (Nguyen et al., 2021)

Method	degree	Hyperparameter					Performance indicator				Time (s)
							R	*RMSE*	*MAE*	*MAPE* (%)	
GEP [51]	1	–	–	–	–	–	0.91	–	5.2	–	–
M-GGP [52]	1	–	–	–	–	–	0.9	7.31	5.48	–	–
ANN-SVR [11]	1	–	–	–	–	–	0.94	6.17	4.24	15.2	–
SFA-LSSVR [12]	1	–	–	–	–	–	0.94	5.62	3.86	12.28	954
MFA-ANN [19]	1	–	–	–	–	–	0.95	4.85	3.41	11.7	276
HO-DNN [20]	1	–	–	–	–	–	0.97	4.05	2.85	–	–
SVR		kernel	C	epsilon	gamma	–					
	1	'rbf'	1000	0.04	0.5	–	**0.95**	**5.00**	**3.79**	**12.73**	**28**
	2	'rbf'	100	0.04	0.4	–	0.95	5.11	3.86	12.98	5
	3	'rbf'	100	0.04	0.3	–	0.95	5.15	3.89	13.06	6
	4	'rbf'	100	0.04	0.3	–	0.95	5.17	3.90	13.04	6
MLP		hidden_layer_sizes	solver	max_iter	alpha	–					
	1	(300, 200)	'lbfgs'	1000	0.0001	–	0.96	4.52	3.19	10.76	136
	2	(100, 300)	'lbfgs'	1000	0	–	0.96	4.39	2.94	9.7	78
	3	(300, 100)	'lbfgs'	1000	0	–	**0.96**	**4.34**	**2.94**	**9.83**	**89**
	4	(100, 300)	'lbfgs'	1000	0	–	0.96	4.44	3.01	10.1	96
GBR		n_estimators	max_depth	learning_rate	loss				min_samples_split		
	1	1000	5	0.1	'huber'	6	**0.97**	**3.77**	**2.44**	**8.31**	**29**
	2	1000	4	0.1	'ls'	6	0.97	3.91	2.57	8.86	26
	3	1000	3	0.1	'huber'	2	0.97	4.04	2.66	9.00	60
	4	1000	3	0.1	'huber'	2	0.97	3.97	2.66	8.95	87
XGBoost		n_estimators	max_depth	learning_rate	objective	–					
	1	1000	4	0.2	'reg:logistic'	–	**0.97**	**3.78**	**2.47**	**8.64**	**5**
	2	1000	4	0.1	'reg:linear'	–	0.97	3.88	2.57	8.89	19
	3	1000	4	0.1	'reg:logistic'	–	0.97	3.97	2.64	8.95	44
	4	1000	3	0.1	'reg:linear'	–	0.97	3.98	2.62	8.86	53

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

Method	# features	degree	Hyperparameter					Performance indicator				Time (s)
								R	*RMSE*	*MAE*	*MAPE* (%)	
Fitting curve [53]	2	1	–	–	–	–	–	0.93	0.45	0.35	15.99	–
MFA-ANN [19]	2	1	–	–	–	–	–	0.96	0.38	0.28	10.59	276
SVR			kernel	C	epsilon	gamma	–					
	2	1	'rbf'	5000	0.03	0.9	–	0.96	0.39	0.27	10.81	9
		2	'rbf'	2000	0.03	0.9	–	**0.96**	**0.38**	**0.27**	**10.64**	**6**
		3	'rbf'	5000	0.03	0.3	–	0.96	0.39	0.27	10.69	7
		4	'rbf'	2000	0.02	0.2	–	0.96	0.39	0.27	10.53	3
	12	1	'rbf'	20	0.01	0.9	–	**0.98**	**0.29**	**0.20**	**7.90**	**2**
		2	'rbf'	10	0.01	0.4	–	0.98	0.29	0.20	7.96	2
		3	'rbf'	10	0.02	0.2	–	0.98	0.29	0.20	8.54	3
		4	'rbf'	10	0.02	0.1	–	0.98	0.29	0.21	8.67	8
MLP			hidden_layer_sizes	solver	max_iter	alpha	–					
	2	1	(300, 300)	'lbfgs'	1000	0.0001	–	0.96	0.39	0.28	10.52	86
		2	(200, 100)	'lbfgs'	400	0.0001	–	0.96	0.38	0.27	10.32	14
		3	(100, 200)	'lbfgs'	1000	0.0001	–	**0.96**	**0.38**	**0.27**	**10.15**	**27**
		4	(200, 100)	'lbfgs'	400	0.0001	–	0.96	0.39	0.27	10.37	9
	12	1	(100, 100)	'lbfgs'	1000	0	–	0.98	0.29	0.20	8.00	26
		2	(300, 300)	'lbfgs'	200	0.0001	–	**0.98**	**0.28**	**0.19**	**8.06**	**35**
		3	(100, 300)	'lbfgs'	300	0.0001	–	0.98	0.28	0.20	8.01	29
		4	(300, 100)	'lbfgs'	100	0.0001	–	0.98	0.29	0.21	8.60	72
GBR			n_estimators	max_depth	learning_rate	loss				min_samples_split		
	2	1	200	3	0.02	'huber'	3	0.96	0.39	0.27	10.68	3
		2	100	3	0.05	'huber'	5	0.96	0.39	0.27	10.58	2
		3	100	3	0.05	'huber'	5	**0.96**	**0.38**	**0.27**	**10.45**	**2**
		4	500	2	0.02	'huber'	3	0.96	0.39	0.27	10.51	5
	12	1	100	4	0.2	'huber'	6	0.98	0.28	0.19	7.06	2
		2	500	3	0.1	'huber'	4	0.98	0.26	0.18	6.89	17
		3	1000	2	0.1	'huber'	6	**0.98**	**0.26**	**0.18**	**6.80**	**94**
		4	500	2	0.1	'huber'	5	0.98	0.28	0.18	7.23	229
XGBoost			n_estimators	max_depth	learning_rate	objective	–					
	2	1	500	4	0.01	'reg:logistic'	–	0.96	0.39	0.28	11.08	1
		2	100	2	0.2	'reg:logistic'	–	**0.96**	**0.38**	**0.28**	**10.67**	**1**
		3	500	3	0.02	'reg:logistic'	–	0.96	0.39	0.28	10.63	2
		4	200	4	0.05	'reg:logistic'	–	0.96	0.39	0.28	10.55	1
	12	1	400	5	0.1	'reg:logistic'	–	0.98	0.28	0.18	7.02	3
		2	1000	4	0.1	'reg:logistic'	–	**0.98**	**0.27**	**0.17**	**6.59**	**22**
		3	1000	2	0.1	'reg:linear'	–	0.98	0.27	0.18	6.97	66
		4	1000	4	0.05	'reg:linear'	–	0.98	0.27	0.18	6.83	544

2.2.8 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 data set 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*.

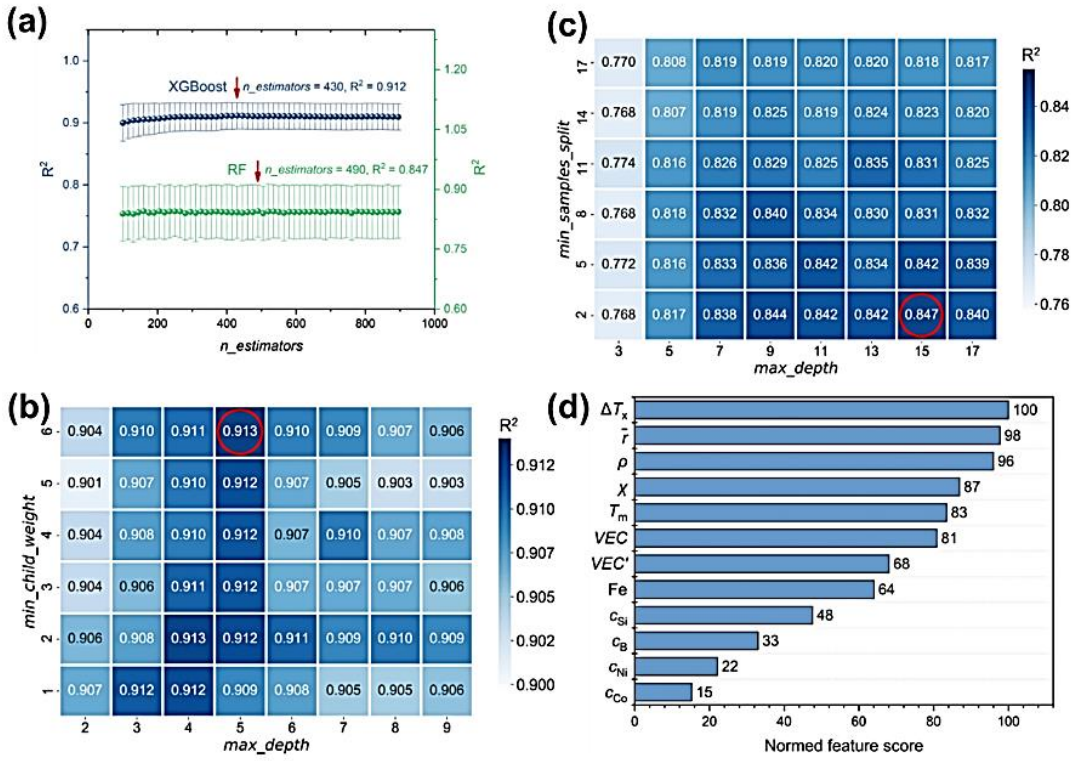
Table 2. 13 hasil *tuning hyperparameter* optimal (Kaushik & Birok, 2021)

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

2.2.9 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 telah dipublikasi, model *machine learning* (ML) dilatih dengan baik untuk memprediksi magnetisasi jenuh (B_s) dari

MG berbasis Fe. *Glass forming ability* diperlakukan sebagai fitur menggunakan data eksperimen dari wilayah cairan super dingin (ΔTx). 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 data set 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 ΔTx 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.



Gambar 2. 11 Hasil penyetelan *hyperparameter* untuk *XGBoost* dan RF dievaluasi oleh R^2. 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).

2.2.10 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 dilihat pada tabel 2.14.

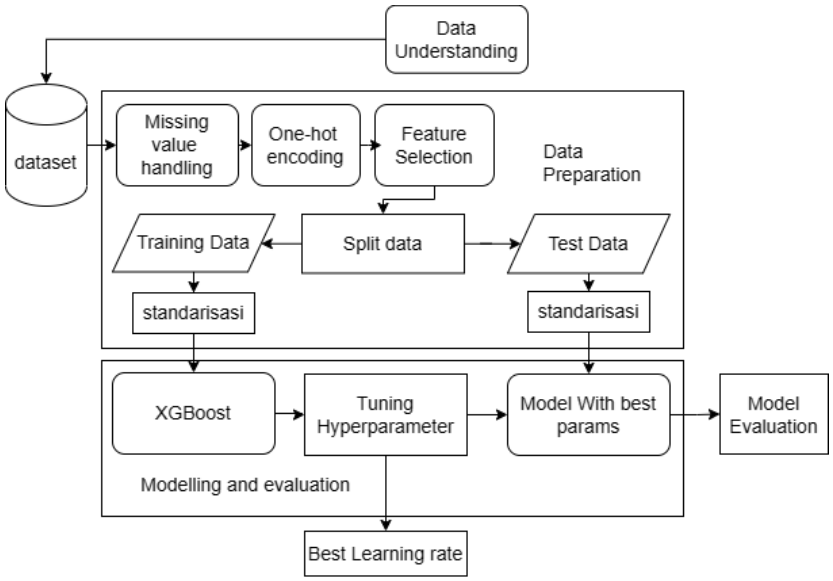
Table 2. 14 Perbandingan performa (Dalal et al., 2022)

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

3.1 Desain Penelitian

Desain pada penelitian ini bertujuan untuk membuat rancangan metode klasifikasi curah hujan dengan menggunakan model *XGBoost*, dan mencari *learning rate* terbaik dari model *XGBoost* yang dibangun dari data set curah hujan. Desain penelitian dapat dilihat pada gambar 3.1.



Gambar 3. 1 Diagram desain penelitian

Tahap awal dalam desain penelitian adalah memahami data apa saja yang akan digunakan untuk membangun model, data tersebut dikumpulkan dalam data set historis. Selanjutnya, dilakukan persiapan data set terlebih dahulu, yaitu berupa penanganan *missing value*, *encoding* fitur kategori, pemilihan fitur, pembagian data latih dan data uji, dan standarisasi data set. Setelah itu, data set dilatih pada model *XGBoost*, dan *tuning hyperparameter* dilakukan untuk mencari *learning rate* terbaik. Terakhir model dievaluasi dengan model default sebagai pembanding apakah model sudah optimal atau belum berdasarkan peningkatan metrik evaluasinya. Masing-masing proses akan dibahas lebih lanjut pada tahapan penelitian.

3.2 Tempat Penelitian

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

3.3 Tahapan Penelitian

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

3.3.1 Identifikasi 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 kejadian hujan yang akan terjadi. Oleh karena itu *XGBoost* diusulkan sebagai model untuk prediksi klasifikasi curah hujan.

3.3.2 Kajian Pustaka

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

3.3.3 *Data Understanding*

Dalam masalah klasifikasi, algoritma pembelajaran mesin mempelajari pola dari data set yang diberikan (Navas, 2022). Data set tersebut berupa variabel yang dapat mempengaruhi variabel target yang akan dilakukan estimasi. Penelitian ini melakukan estimasi terhadap data kategori curah hujan, atau klasifikasi curah hujan, di mana 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 dan arah 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 data set 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 magnetik, arah angin sejati, kecepatan angin, dan curah hujan. Data secara keseluruhan dapat dilihat pada table 3.1

Tabel 3. 1 Tabel data set yang digunakan dalam penelitian.

No	***Date and Time***	***Air Tmp (℃) 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	0.0	1009.01	100.0	84.0	2.0
1	01/12/2022 01:00	26.51	61.0	0.0	1009.35	92.9	63.0	2.0
2	01/12/2022 02:00	28.45	3.0	0.0	1009.47	81.8	5.0	2.0
3	01/12/2022 03:00	29.59	340.0	0.0	1009.11	75.2	342.0	2.0
4	01/12/2022 04:00	30.68	297.0	0.0	1008.14	69.8	299.0	4.0
...	...	...	...	...	...	...	...	...
2039	23/02/2023 23:00	24.16	331.0	NaN	1009.32	92.5	333.0	3.0

Berdasarkan data tabel 3.1 dapat 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 data set, kuartil 1 (25%), kuartil 2 (50%), kuartil 3 (75%) dan nilai *maximum* (*max*) dari data set penelitian. Secara rinci ditampilkan pada tabel 3.2.

Tabel 3. 2 Statistik Data set

	Air Tmp (℃) 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

Dari nilai *count* pada tabel 3.2 dapat dilihat bahwa total data tidak sama untuk masing-masing fitur. Total data pada tabel 3.1 adalah 2040, sedangkan total yang tercatat pada tabel 3.2 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 3.2 terdapat 7 variabel, yaitu *Date and Time, Air tmp (℃) 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 variabel dan nilai *missing value*-nya ditampilkan pada tabel 3.3.

Tabel 3. 3 Penjelasan masing-masing variabel dan nilai yang hilang

No	Fitur	Deskripsi	Total data yang hilang	Satuan
1	*Date and Time*	Tanggal dan waktu per jamnya	135	Jam
2	*Air tmp (℃) 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 kategori hujan, yaitu ketika data curah hujan lebih besar dari 0 dan kategori tidak hujan, ketika data curah hujan sama dengan 0. Sedangkan *multiclass classification* akan dikelompokkan sesuai dengan standar internasional *World Meteorogical Organization* (WMO) tahun 2008 seperti ditampilkan pada tabel 3.4.

Tabel 3. 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 di luar klasifikasi WMO

Tabel 3.4 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, sehingga terdapat 3 kategori pada *multiclass classification*, seperti yang ditampilkan pada tabel 3.5.

Tabel 3. 5 Tabel klasifikasi data secara keseluruhan

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

Data set secara keseluruhan ditampilkan pada tabel 3.6

Tabel 3. 6 Keseluruhan data set

No	***Date and Time***	***Air Tmp (℃) 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
...	...	...	...	...	...	...	...	...
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

3.3.4 *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 data set menjadi data latih dan uji, standarisasi data latih dan data uji.

3.3.4.1 Penanganan *missing value*

Data yang hilang dapat mempengaruhi proses pelatihan. Terdapat beberapa metode yang digunakan untuk menangani *missing value*, di antaranya 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 historisnya. Nilai rata-rata yang diambil adalah nilai rata-rata dua data, yaitu nilai sebelum dan setelah *missing value.* Seperti yang di ilustrasikan pada gambar 3.2

22,49	0
24,77	0

Gambar 3. 2 Ilustrasi penaganan *missing value*

Nilai *missing value* pada tabel tersebut akan diisi dengan 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 per jam, maka data satu jam setelahnya atau beberapa jam setelahnya kemungkinan besar tidak sejauh nilai sebelum dan sesudah *missing value* tersebut.

3.3.4.2 *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 ke dalam 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 vektor, dengan nilai 0 dan 1. Setiap bit vektor ini merepresentasikan satu kemungkinan nilai,

artinya panjang dari vektor ini sama dengan jumlah dari kemungkinan nilai atau jumlah kategori yang ada (Erjavac et al., 2022). Hasil *encoding* fitur kategori dapat dilihat pada tabel 3.7.

Tabel 3. 7 *one-hot encoding* fitur kategori pada data set

Binary classification								
Date and Time	**Air Tmp (℃) 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 (℃) 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
...	...	...	...	...	...	...	...	...

3.3.4.3 Pemilihan Fitur

Data fitur independen 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 pada data *input* tidak digunakan karena memiliki korelasi rendah terhadap kejadian hujan dan memiliki skor fitur penting terendah dalam model, secara lengkap dijelaskan pada analisis data. Sedangkan fitur dependen atau target adalah data kategori yang sudah dilakukan *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*.

3.3.4.4 Pembagian data set

Dalam proyek ini, data set 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 data set 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 data set tersebut, digunakan modul *train_test_split* dari *scikit-learn*. Sehingga, jumlah sampel yang digunakan pada model dapat dilihat pada tabel 3.8.

Tabel 3. 8 Tabel pembagian data set *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

3.3.4.5 Standarisasi data set

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 3.9.

Tabel 3. 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

3.4 Implementasi Model *XGBoost*

Model *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 *modeling*, model akan dilatih dengan default *hyperparameter* terlebih dahulu sebagai pembanding apakah model sudah optimal atau belum.

3.4.1 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 3.10. Nilai ini akan diimplementasikan pada fungsi *XGBClassifier* yang terdapat dalam modul *XGBoost* untuk menangani klasifikasi. Secara rinci bisa dilihat dalam gambar 3.3.

Tabel 3. 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

```
xgb = XGBClassifier(
    max_depth=6,
    n_estimators=100,
    learning_rate=0.3, )
xgb.fit(X_train, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.3, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=6, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

Gambar 3. 3 Gambar model *XGBClassifier* secara default

3.4.2 Tuning *hyperparameter*

Hyperparameter yang akan dicari pada penelitian ini adalah nilai *learning rate*. Pencarian menggunakan 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.

```
from scipy.stats import uniform, randint
import numpy as np
params = {
    'n_estimators': [100],
    'max_depth': [6],
    'learning_rate': uniform(0.01, 1)
}
```

Gambar 3. 4 Inisialisai rentang pencarian *hyperparameter*

Pada gambar 3.4, nilai *learning rate* pada pencarian secara kasar adalah dalam rentang 0,01 hingga 1, berupa bilangan acak 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 3.11, dan implementasinya pada gambar 3.5.

Tabel 3. 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*

```
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)
```

Gambar 3. 5 Implementasi kode pada *RandomSearchCV*

Nilai *hyperparameter* optimal *learning rate* diambil dari *search.best_params_* dari modul *sklearn*, dan nilai ini disimpan dalam variabel *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 3.6.

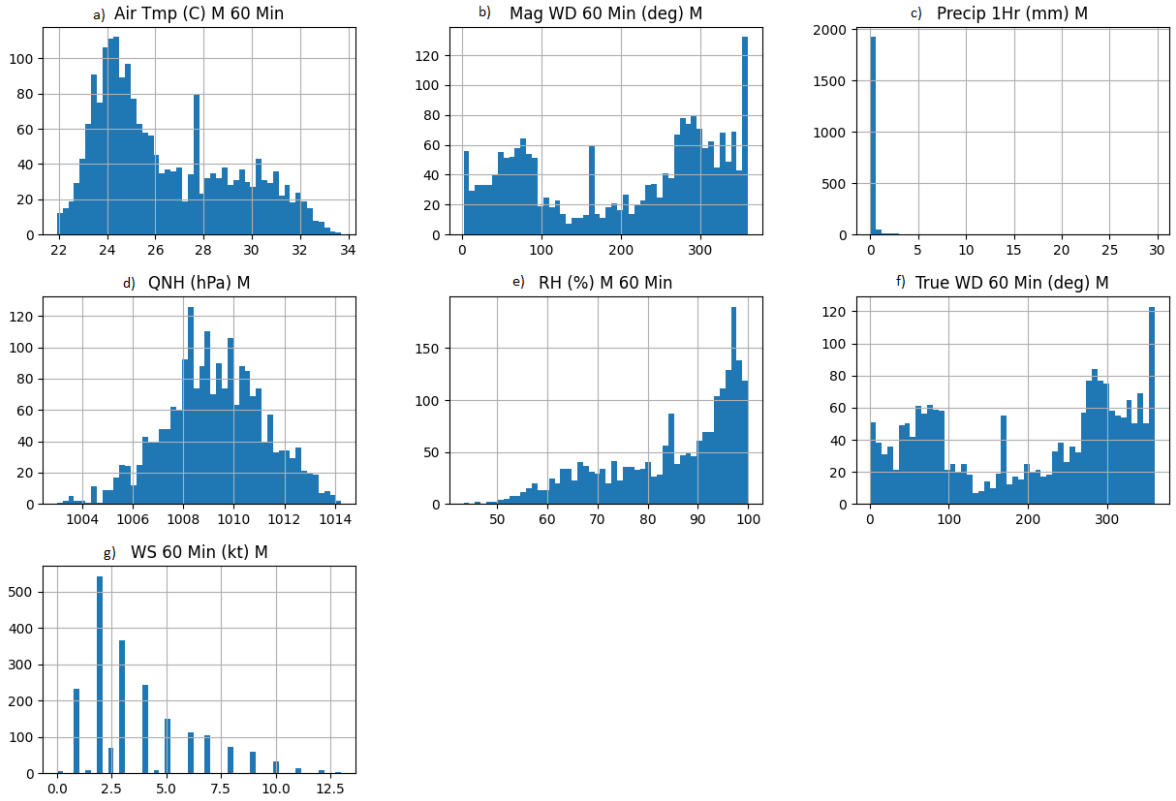
```
best_params = search.best_params_
param_dist = {
    'n_estimators': [100],
    'max_depth': [6],
    'learning_rate': np.logspace(
        np.log10(best_params['learning_rate']-0.01),
        np.log10(best_params['learning_rate']+0.01), 50
        ),
}

fine_search = GridSearchCV(
    xgb_model, param_grid=param_dist,
    cv=10,
    scoring='accuracy',
    n_jobs=-1,
    )
fine_search.fit(X_train, y_train)
```

Gambar 3. 6 Implementasi *GridSearchCV*

BAB 4
HASIL DAN PEMBAHASAN

4.1 Analisis Data

Analisis data dilakukan untuk melihat pola data dan memahami karakteristik data. Dalam *univariate* data analisis, data masing-masing variabel diamati dengan menggunakan *bar plot.* Sebaran data numerik dapat dilihat pada gambar 4.1.

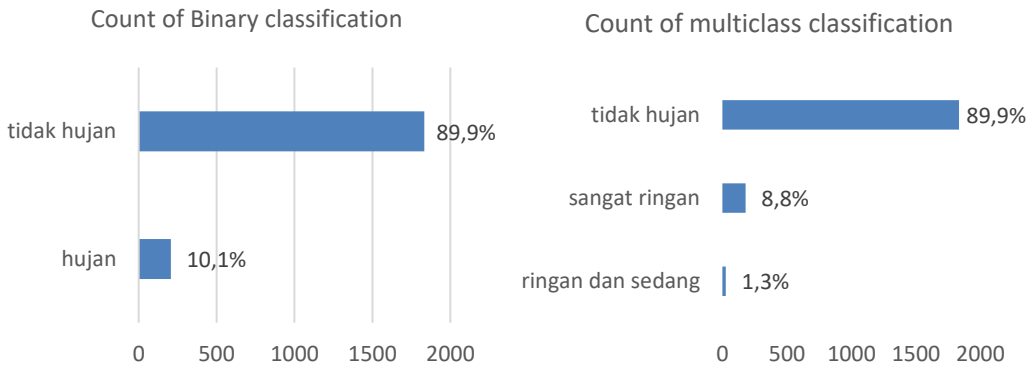


Gambar 4. 1 *Bar plot* data numerik, a) suhu ℃, b) arah angin magnetic (deg), c) curah hujan (mm), d) tekanan atmosfer (hPa), e) kelembapan (%), arah angin sejati (deg), g) kecepatan angin (kt)

Pola data numerik pada data set *time series* memiliki pola tertentu, sehingga model mempelajari pola data ini dalam melakukan estimasi. Data suhu berubah-ubah sesuai dengan musim yang terjadi di Kota Pontianak. Berdasarkan sampel data yang diambil, data suhu 22 hingga 27℃ memiliki intensitas lebih tinggi dibandingkan data suhu lainnya. Sedangkan data

curah hujan memiliki data yang tidak seimbang atau *imbalance*, di mana nilai nol mendominasi keseluruhan data. Dengan data yang tidak seimbang metode *forecasting* atau metode regresi yang menggunakan data numerik sebagai target akan sulit dilakukan, sehingga untuk melakukan estimasi, metode klasifikasi merupakan pilihan yang tepat (Depto et al., 2023). Selanjutnya pada tekanan atmosfer, mayoritas data memiliki nilai yang normal yaitu antara 1000 hingga 1014. Kemudian pada data kelembapan, jumlah sampel dengan kelembapan tinggi semakin meningkat seiring meningkatnya kelembapan udara, hal ini dapat dilihat jelas pada grafik yang mengalami peningkatan. Sedangkan kecepatan angin didominasi oleh angin dengan kecepatan rendah yaitu 1-6 kt. Untuk arah angin magnetik dan arah angin sejati, sebaran jumlah data tidak memiliki pola tertentu, karena jumlah data dan arah angin merata pada nilai tinggi dan rendah.

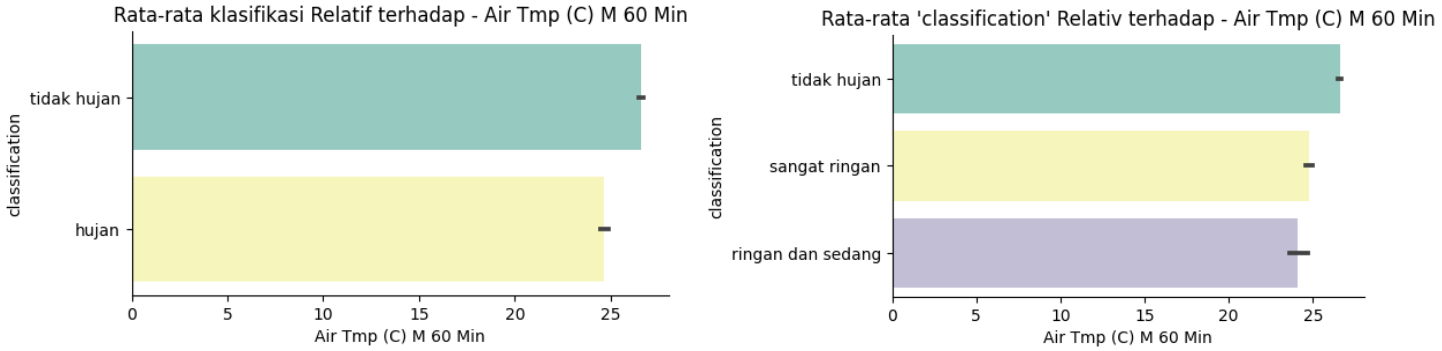
Selanjutnya sebaran data dan persentase pada data kategori dapat dilihat pada gambar 4.2. sebagai berikut:



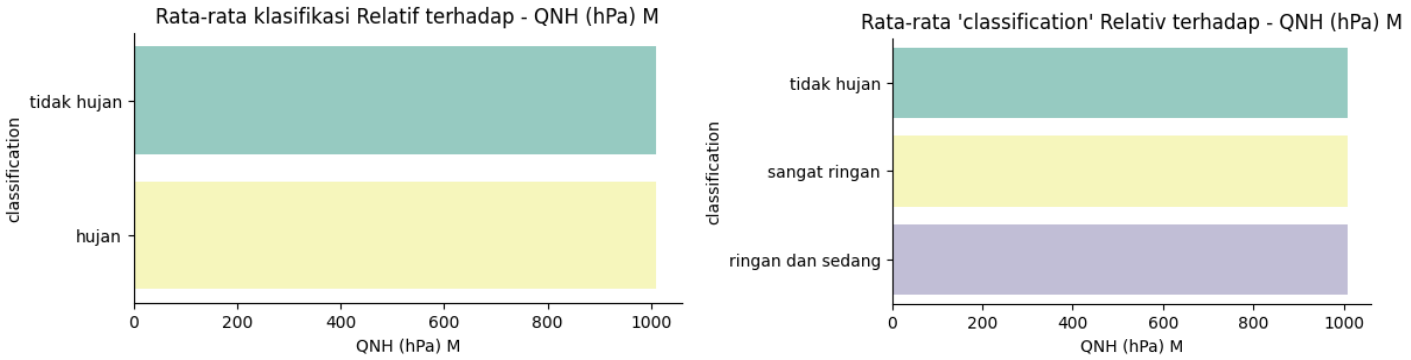
Gambar 4. 2 *Bar plot* data kategori untuk klasifikasi *binary* dan *multiclass.*

Persentase data paling tinggi adalah pada kategori tidak hujan yaitu sekitar 89.9%, sedangkan persentase kategori lainnya bernilai sangat kecil, sehingga model lebih cenderung memahami data tidak hujan dibandingkan dengan kategori lainnya.

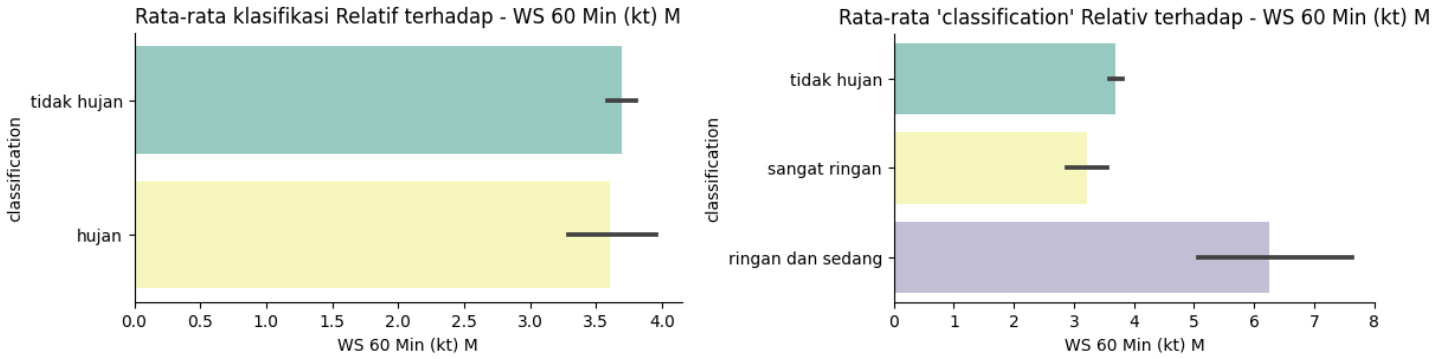
Selanjutnya analisis data yang cukup penting adalah melihat korelasi atau keterikatan antara masing-masing variabel data atau banyak variabel, atau disebut juga analisis *multivariate.* Data kategori dan korelasinya dengan masing-masing data numerik dapat dilihat gambar 4.3 dan 4.4.



a)

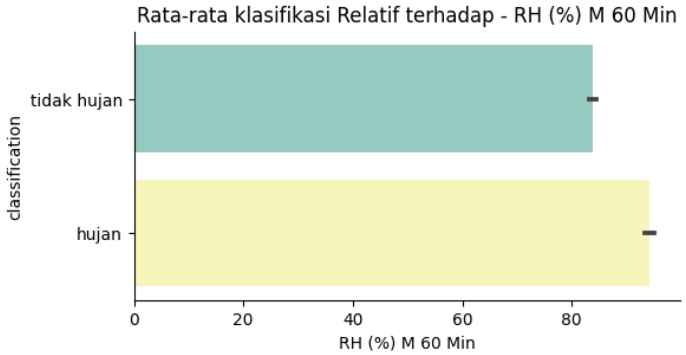


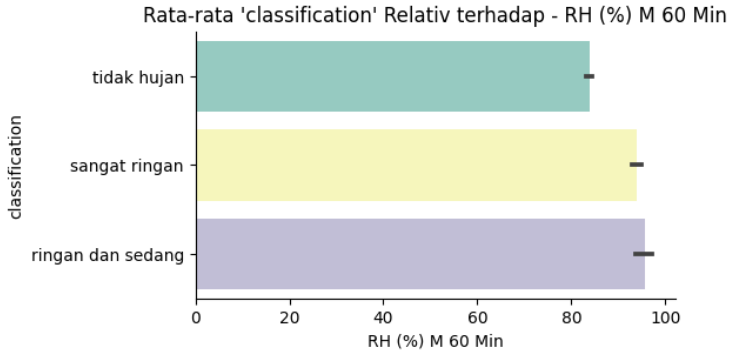
b)



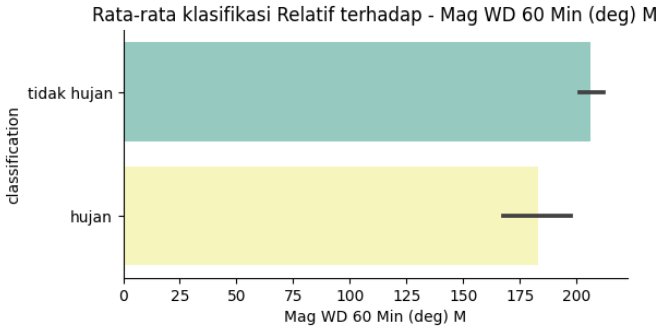
c)

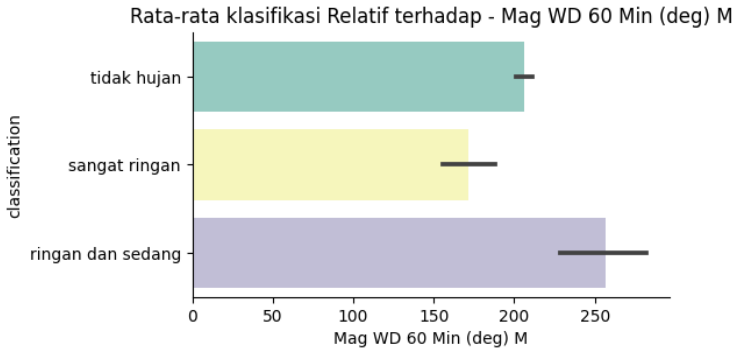
Gambar 4. 3 a) rata-rata klasifikasi biner dan *multiclass* terhadap suhu, b) rata-rata klasifikasi biner dan *multiclass* terhadap tekanan atmosfer, c) rata-rata klasifikasi biner dan *multiclass* terhadap kecepatan angin.



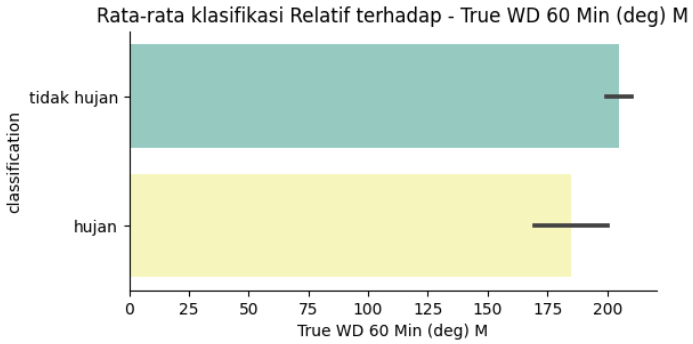


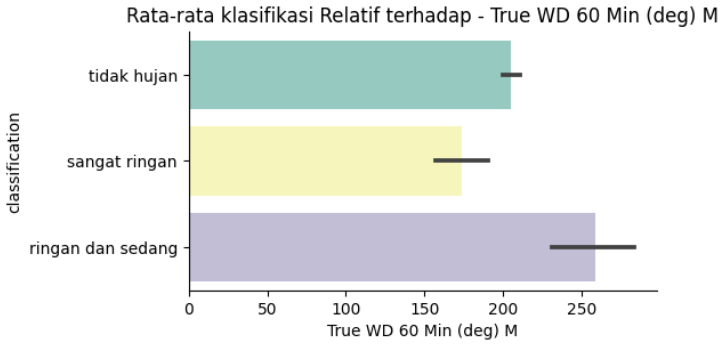
a)





b)





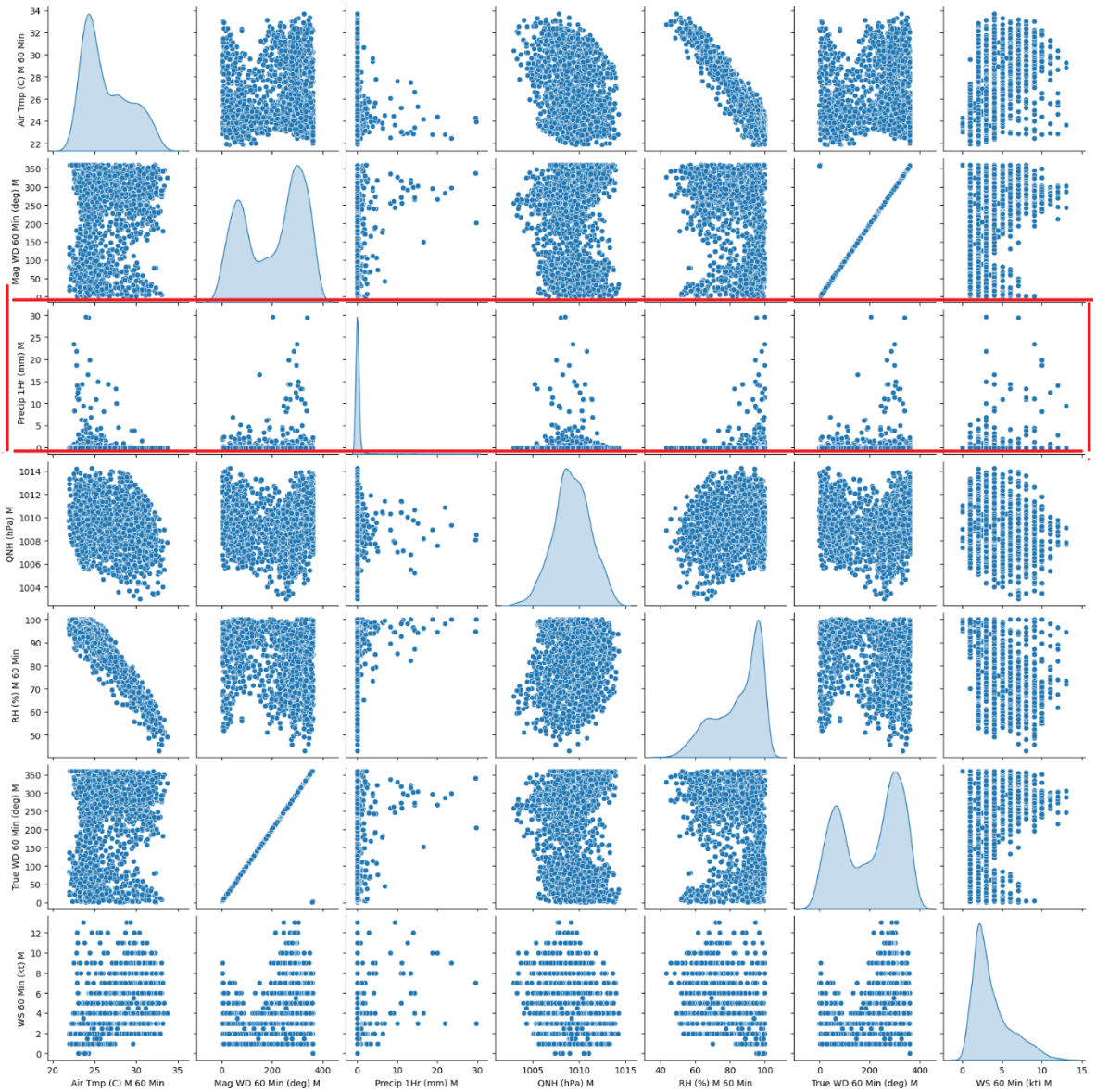
c)

Gambar 4. 4 a) rata-rata klasifikasi biner dan *multiclass* terhadap kelembapan, b) rata-rata klasifikasi biner dan *multiclass* terhadap arah angin magnetic, c) rata-rata klasifikasi biner dan *multiclass* terhadap arah angin sejati

Dari gambar 4.3 dan 4.4 dapat diamati rata-rata relatif fitur independen terhadap data kategori adalah sebagai berikut:

1. Pada fitur suhu udara, kejadian hujan baik ringan dan sedang maupun sangat ringan, terjadi ketika suhu berada di bawah 25^0C, sedangkan kategori tidak hujan berada pada suhu di atas 25^0C.
2. Fitur tekanan atmosfer memiliki rata-rata nilai yang sama untuk masing-masing data kategori.
3. Pada fitur kecepatan angin untuk klasifikasi biner, rata-rata kecepatan angin lebih rendah ketika hujan dibandingkan dengan ketika tidak hujan. Sedangkan pada klasifikasi *multiclass*, rata-rata kecepatan angin lebih tinggi ketika kejadian hujan ringan dan sedang, dibandingkan dengan dua kategori lainnya.
4. Pada fitur kelembapan, nilai rata-rata kelembapan pada kejadian hujan lebih tinggi daripada kejadian tidak hujan.
5. Pada fitur arah angin magnetik dan arah angin sejati untuk klasifikasi biner, rata-rata kecepatan angin lebih rendah ketika hujan dibandingkan dengan ketika tidak hujan. Sedangkan pada klasifikasi *multiclass*, rata-rata kecepatan angin lebih tinggi ketika kejadian hujan ringan dan sedang, dibandingkan dengan dua kategori lainnya. Hal ini sama dengan variabel kecepatan angin

Selanjutnya, pada fitur numerik dapat diamati korelasi atau hubungan masing-masing variabel menggunakan grafik *pairplot* seperti pada gambar 4.5.

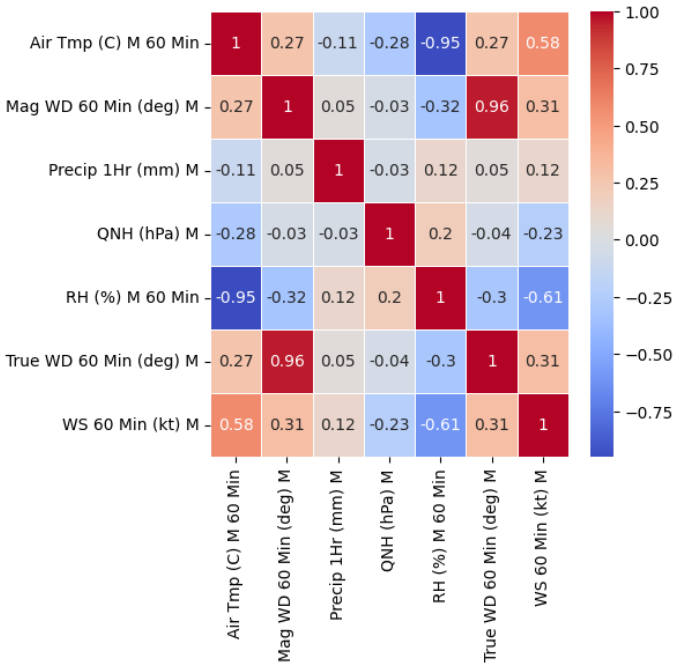


Gambar 4. 5 Grafik *pairplot* data numerik masing-masing variabel.

Dengan mengamati grafik *pariplot*, dapat dilihat korelasi masing-masing fitur, apakah grafik sebaran data memiliki pola (memiliki korelasi), sebaran data acak (korelasi lemah) atau tidak memiliki pola (tidak memiliki korelasi). Pada kasus ini, variabel yang harus diamati adalah *precip 1Hr (mm) M* (dalam kotak merah), yang merupakan nilai curah hujan yang menjadi dasar dalam menentukan kategori hujan sebagai variabel target model, secara rinci hubungan curah hujan dengan variabel lain dapat dilihat pada lampiran 4. Pada gambar 4.5, dapat dilihat bahwa sebagian besar data pada variabel curah hujan terpusat pada nilai

nol, dan sebagian kecil memiliki pola. Pada data suhu udara *(Air Tmp (^{0}C) M 60 Min)* dengan sebagian kecil yang memiliki pola, dapat diamati bahwa semakin tinggi suhu udara curah hujan semakin kecil. Sedangkan pada data tekanan atmosfer *(QNH (hPa) M)* dan kecepatan udara *(WS 60 Min (kt) M)* terlihat sebagian data memiliki pola yang acak, sehingga memiliki korelasi yang lemah. Selanjutnya pada data kelembapan *(RH (%) M 60 Min)*, sebagian data menunjukkan bahwa semakin tinggi kelembapan udara, curah hujan juga semakin tinggi. Kemudian pada data arah angin magnetik dan arah angin sejati, dapat dilihat sebagian kecil sebaran data membentuk pola, di mana arah angin yang besar menunjukkan nilai curah hujan yang tinggi. Sehingga dapat disimpulkan, variabel independen memiliki korelasi terhadap data target.

Korelasi variabel independen dengan data curah hujan dapat ditampilkan skor korelasinya dalam bentuk matriks korelasi, seperti yang ditampilkan pada gambar 4.6



Gambar 4. 6 Matriks korelasi masing-masing fitur.

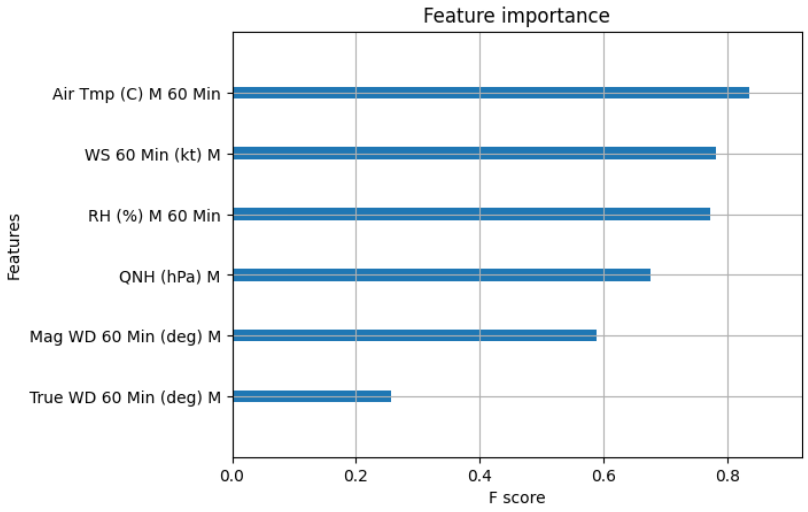
Skor korelasi menunjukkan kekuatan keterikatan antar dua fitur yang diamati. Semakin mendekati 1 atau -1, korelasi semakin kuat. Jika mendekati -1 artinya data tersebut berkorelasi negatif atau berbanding terbalik, dan jika mendekati 1 maka data tersebut

memiliki korelasi positif. Korelasi yang akan diamati adalah data curah hujan dengan variabel lainnya, seperti yang ditampilkan pada tabel 4.1.

Tabel 4. 1 Tabel korelasi curah hujan dengan variabel independen

	Precip 1Hr (mm) M
Precip 1Hr (mm) M	1.00
WS 60 Min (kt) M	0.12
RH (%) M 60 Min	0.12
True WD 60 Min (deg) M	0.05
Mag WD 60 Min (deg) M	0.05
QNH (hPa) M	-0.03
Air Tmp (C) M 60 Min	-0.11

Dapat dilihat pada tabel bahwa curah hujan memiliki korelasi yang rendah terhadap variabel independen. Korelasi yang rendah ini disebabkan oleh ketidakseimbangan data curah hujan, di mana data kategori hujan lebih sedikit dibandingkan dengan data tidak hujan, sehingga nilai curah hujan mayoritas adalah nol. Selain itu, variabel tekanan atmosfer dan suhu memiliki korelasi negatif dan variabel kecepatan angin dan kelembapan memiliki korelasi positif. Hal ini sesuai dengan penjelasan pemahaman data pada poin 3.3.3. Selanjutnya, variabel-variabel ini dapat dilihat nilai *gain* yang diperoleh dalam membangun *decision tree* model, seperti yang ditampilkan pada gambar 4.7.



Gambar 4. 7 plot fitur penting pada model

Sebagaimana dapat dilihat pada gambar 4.7, arah angin sejati dan arah angin magnetik memiliki gain yang kecil, sehingga kedua variabel arah angin tidak digunakan dalam variabel independen atau *input*, karena semakin tinggi *gain* (sumbu x), semakin penting fitur tersebut dalam pembangunan model.

4.2 Hasil Tuning *Hyperparameter*

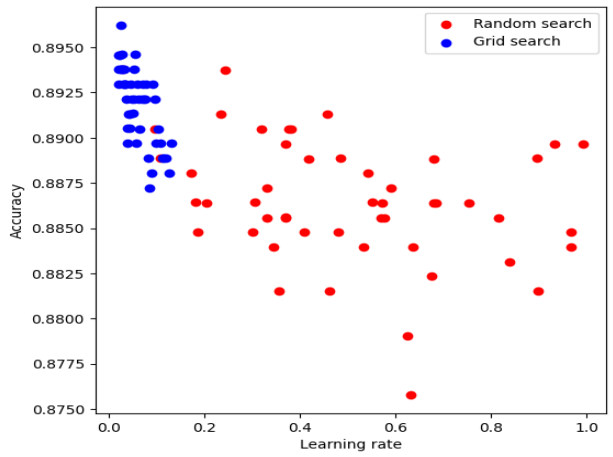
Pada penelitian ini nilai *hyperparameter* yang dicari adalah nilai *learning rate* dengan menggunakan teknik *coarse to fine*. Pencarian *hyperparameter* dilakukan dengan menganalisis masing-masing pembagian data *train* dan data *test*. Hasil yang terbaik akan dipertimbangkan berdasarkan nilai akurasi pada data latih. Hasil *tuning* dapat dilihat pada tabel 4.2.

Tabel 4. 2 Hasil *tuning hyperparameter*

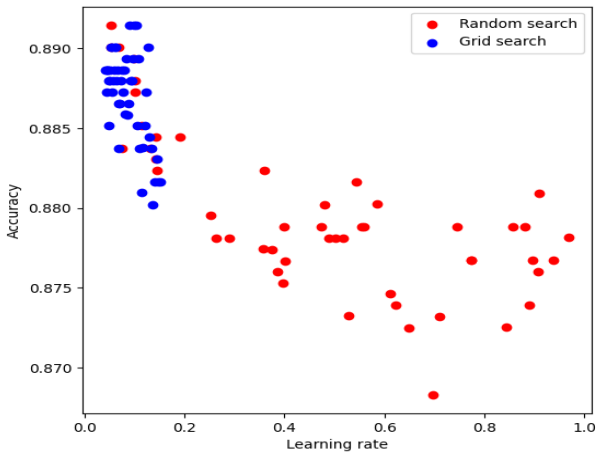
Split train-test	*Best learning rate*		*Accuracy*	
	multiclass	*Binary*	*multiclass*	*Binary*
90%-10%	0,030	0,026	89,76%	90,14%
80%-20%	0,042	0,047	89,21%	90,25%
70%-30%	0,029	0,089	88,51%	89,15%
60%-40%	0,036	0,024	88,89%	89,62%

Pada Tabel 4.2 dapat dilihat bahwa nilai *accuracy* data latih tertinggi untuk *binary classification* adalah pada *split train* dan *test* sebesar 80%-20%, sebesar 90.25% dan *learning rate* terbaik adalah 0,047. Sedangkan pada klasifikasi *multiclass accuracy* terbaik yang diperoleh sebesar 89.76% dan nilai *learning rate* sebesar 0.030. Nilai *learning rate* ini diperoleh berdasarkan model yang dibangun dari data set, sehingga nilai ini bersifat dependen atau bergantung pada *decision tree* masing-masing model. Oleh karena itu dengan nilai *learning rate* yang sama atau mendekati sama seperti pada pembagian klasifikasi *binary* 90%-10% dan 60%-40% yaitu 0,026 dan 0,024 memiliki nilai akurasi yang cukup berbeda. Hal ini disebabkan karena proporsi data set pada pembagian 90%-10% lebih besar dibanding 60%-40%, sehingga model yang dibangun lebih kompleks dan detail, dan membuat akurasi dari model lebih tinggi. Hasil pencarian *learning rate* dapat divisualisasikan pada grafik seperti pada gambar 4.8 dan 4.9.

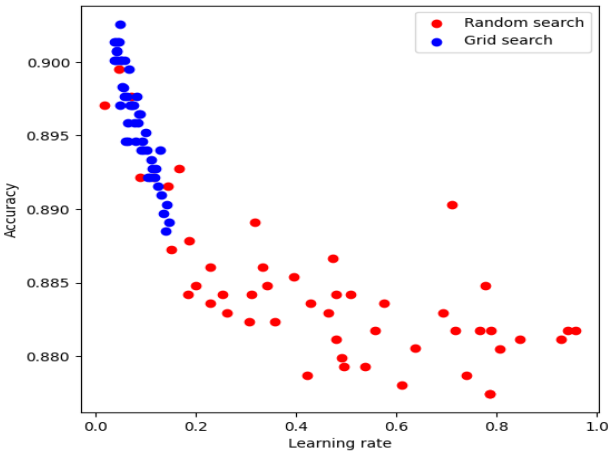
a)



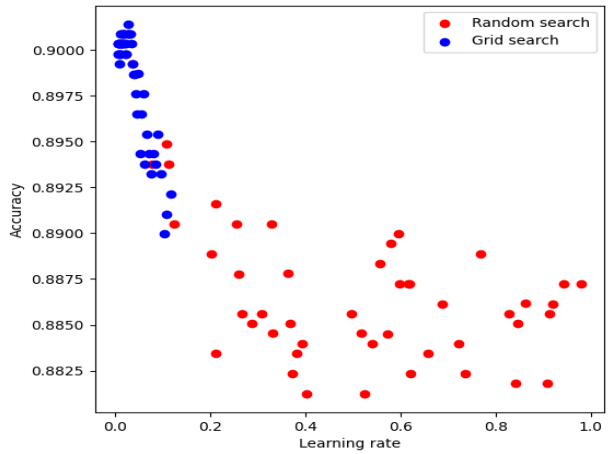
b)



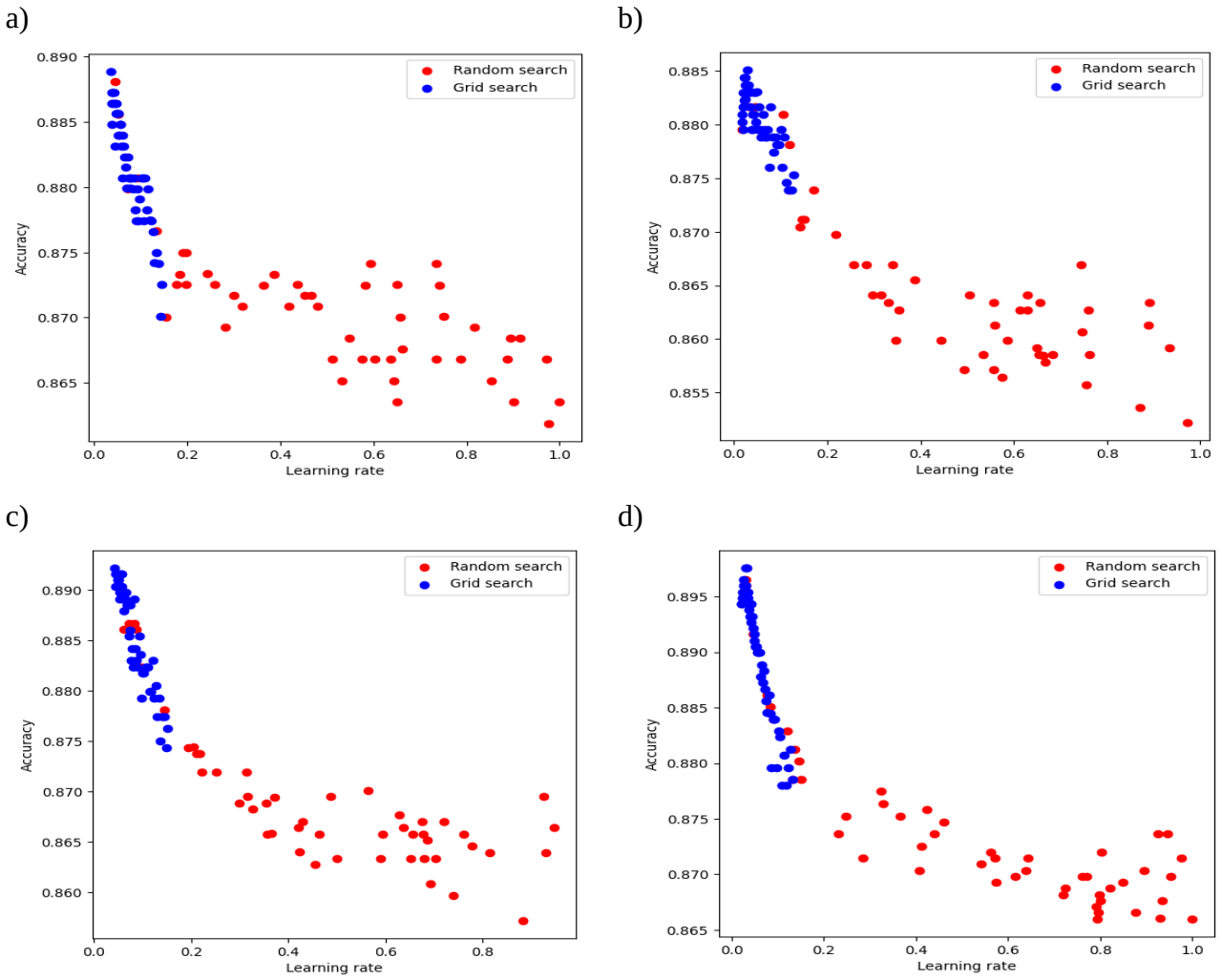
c)



d)



Gambar 4. 8 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%



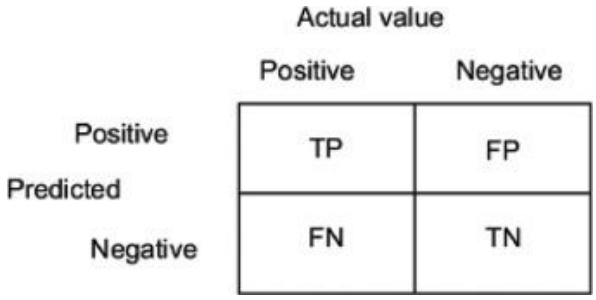
Gambar 4. 9 Visualisasi pencarian *learning rate* dengan teknik *coarse to fine*. a) *multiclass* 60%-40%, b) *multiclass* 70%-30%, c) *multiclass* 80%-20%, d) *multiclass* 90%-10%

Berdasarkan gambar 4.8 dan 4.9, dapat diperhatikan bahwa nilai *learning rate* yang memiliki akurasi tinggi berada pada rentan 0,0 hingga 0,1. Selain itu, pada data set 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.

4.3 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. Hasil evaluasi data uji dapat diamati menggunakan *confusion matrix*, yaitu

sebuah konsep dari pembelajaran mesin yang menunjukkan informasi tentang nilai aktual dan nilai prediksi yang telah dilakukan oleh model (Deng et al., 2016). Secara sederhana ilustrasi *confusion matrix* dapat dilihat pada gambar 4.10.



Gambar 4. 10 *confusin matrix* (Canayaz, 2021).

Dengan mengamati *confusion matrix*, performa model secara keseluruhan dapat dihitung menggunakan nilai *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negeative* (TN). Dalam klasifikasi biner, data positif merupakan data hujan dan data negatif adalah data tidak hujan, sehingga TP berarti prediksi benar untuk data hujan, FP adalah prediksi salah untuk data hujan, FN adalah prediksi salah untuk data tidak hujan, dan TN adalah prediksi benar untuk data tidak hujan. Dari nilai-nilai ini, ada 4 performa model yang dihitung yaitu akurasi (*Acc*), presisi (*Pre*), *recall* (*Re*), dan f1-score (*F1-scr*). Secara matematis dituliskan sebagai berikut (Canayaz, 2021):

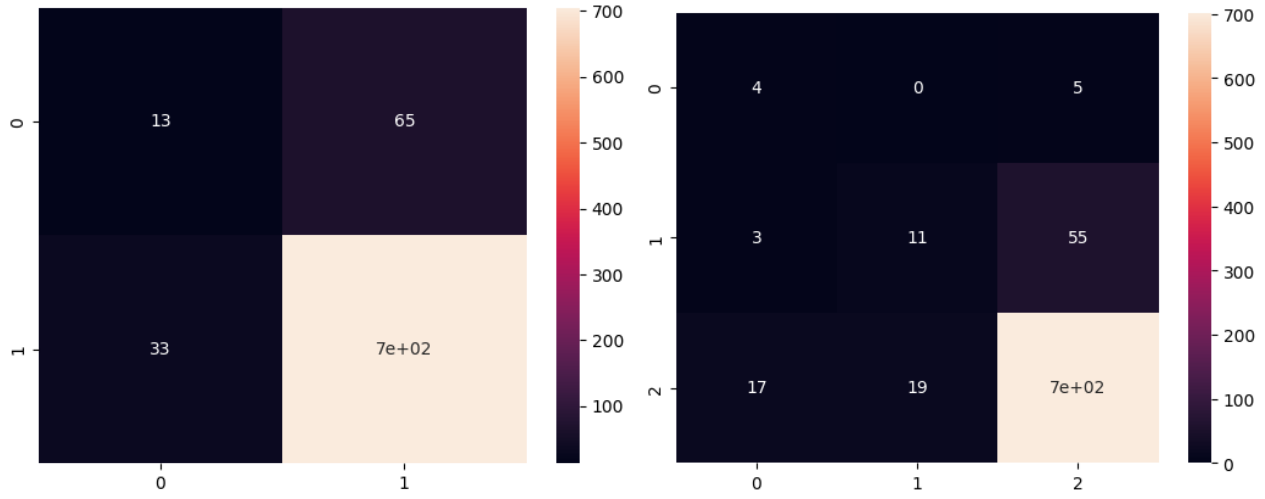
$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$Pre = \frac{TP}{TP + FP} \tag{4.2}$$

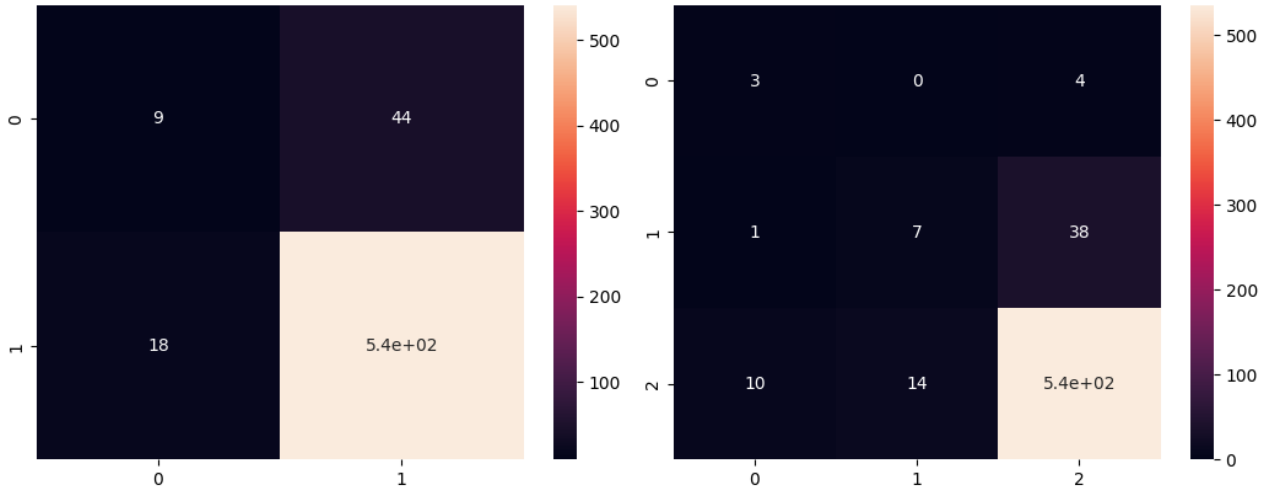
$$Re = \frac{TP}{TP + FN} \tag{4.1}$$

$$F1 - scr = \frac{2TP}{2TP + FP + FN} \tag{4.1}$$

Secara keseluruhan, c*onfusion matrix* pada klasifikasi biner dan *multiclass* dapat dilihat pada gambar 4.11 dan 4.12.

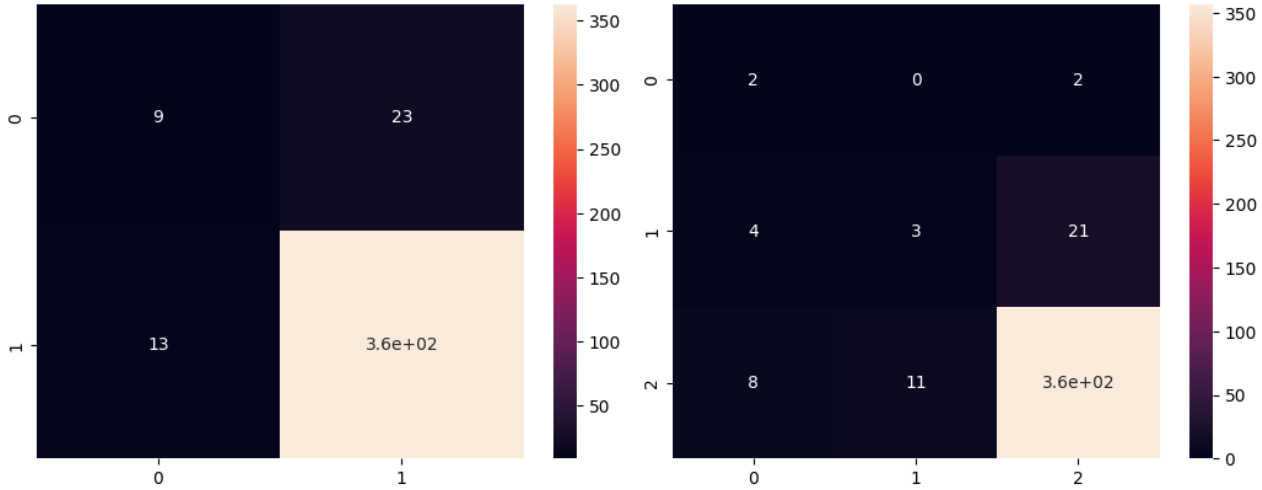


a)

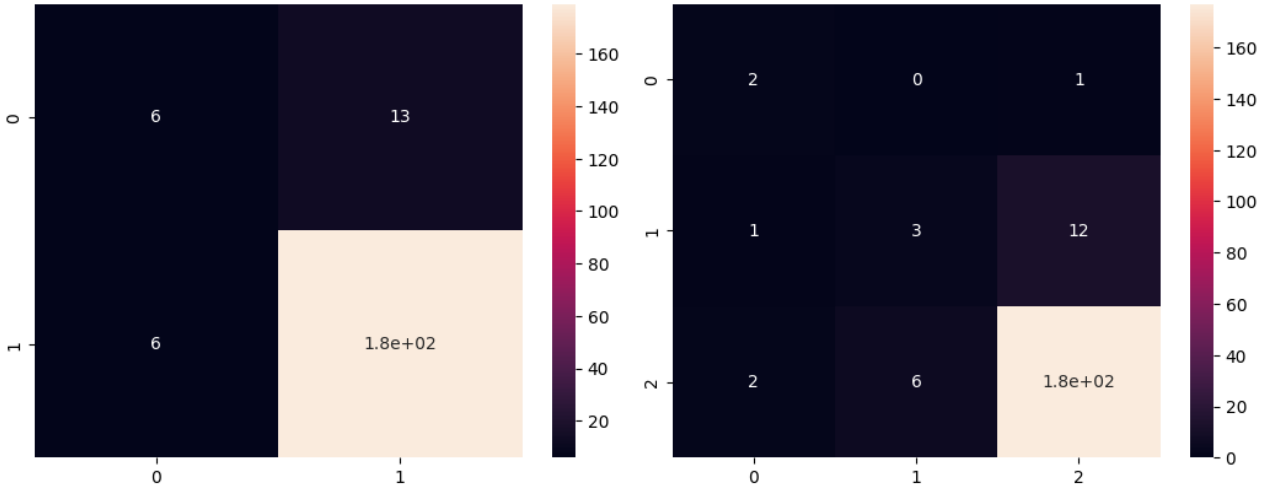


b)

Gambar 4. 11 klasifikasi biner pada bagian kiri dan *multiclass* pada bagian kanan, a) klasifikasi biner dan *multiclass* pada kasus 60-40 pembagian data, b) klasifikasi biner dan *multiclass* pada kasus 70-30 pembagian data,



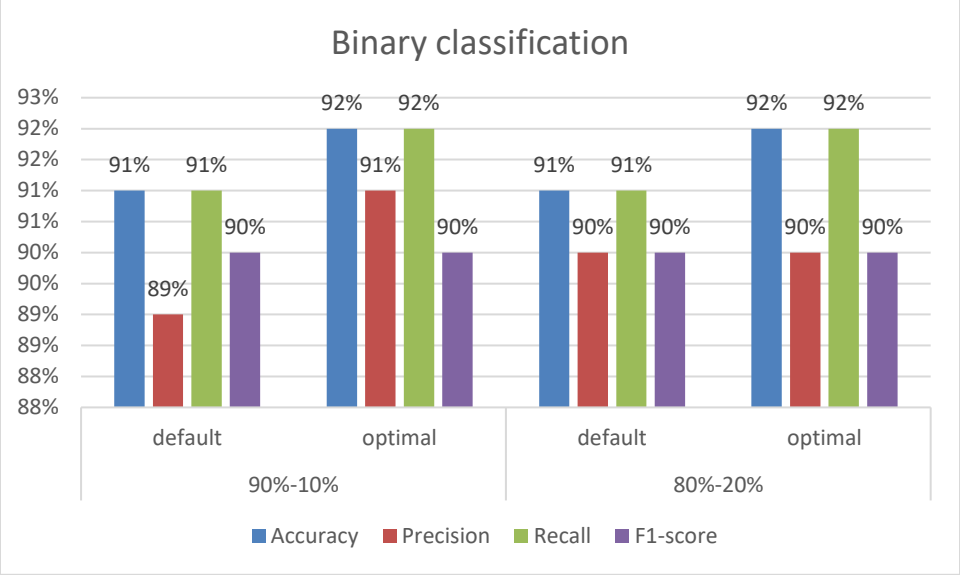
a)



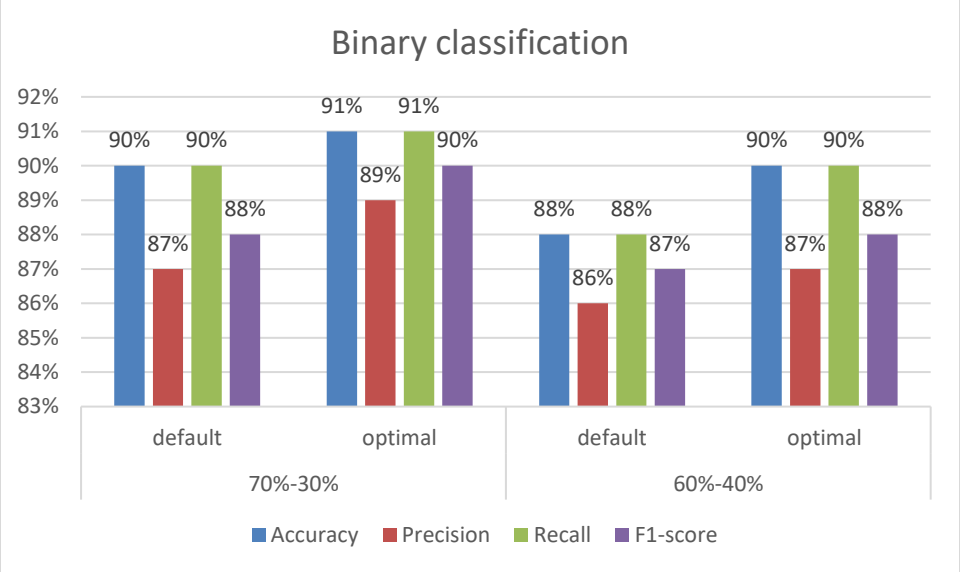
b)

Gambar 4. 12 klasifikasi biner pada bagian kiri dan *multiclass* pada bagian kanan, a) klasifikasi biner dan *multiclass* pada kasus 80-20 pembagian data, b) klasifikasi biner dan *multiclass* pada kasus 90-10 pembagian data.

Untuk memudahkan penghitungan performa model, modul *sklearn.metrics* telah menyediakan *library* yang dapat melakukan penghitungan performa dengan mudah, yaitu *accuracy_score*, *precision_score*, *recall_score*, dan *f1_score*. Nilai evaluasi masing-masing klasifikasi dapat dilihat pada Diagram batang gambar 4.13 dan 4.14.

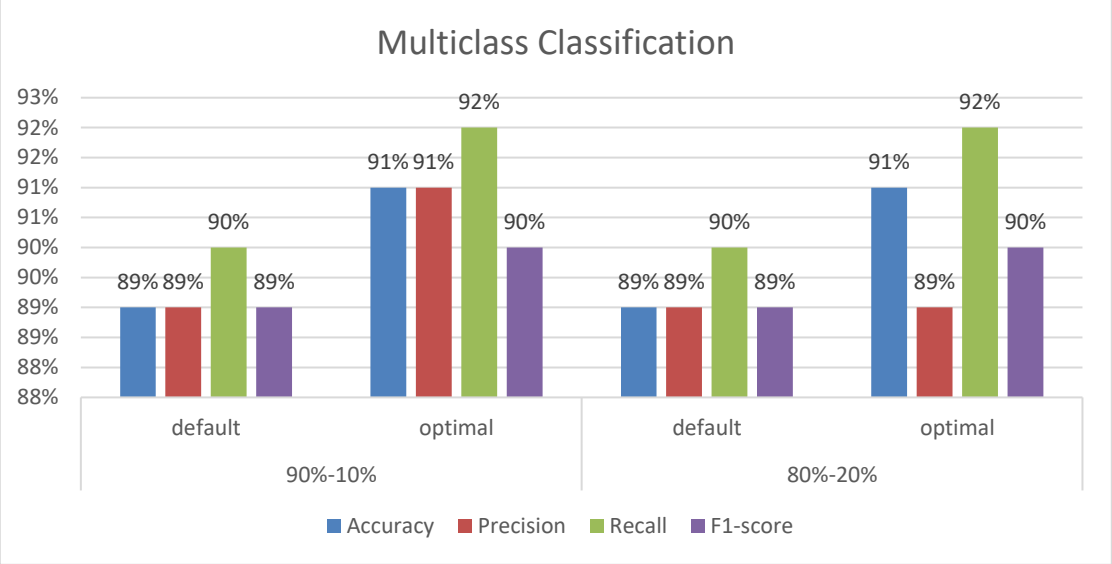


a)

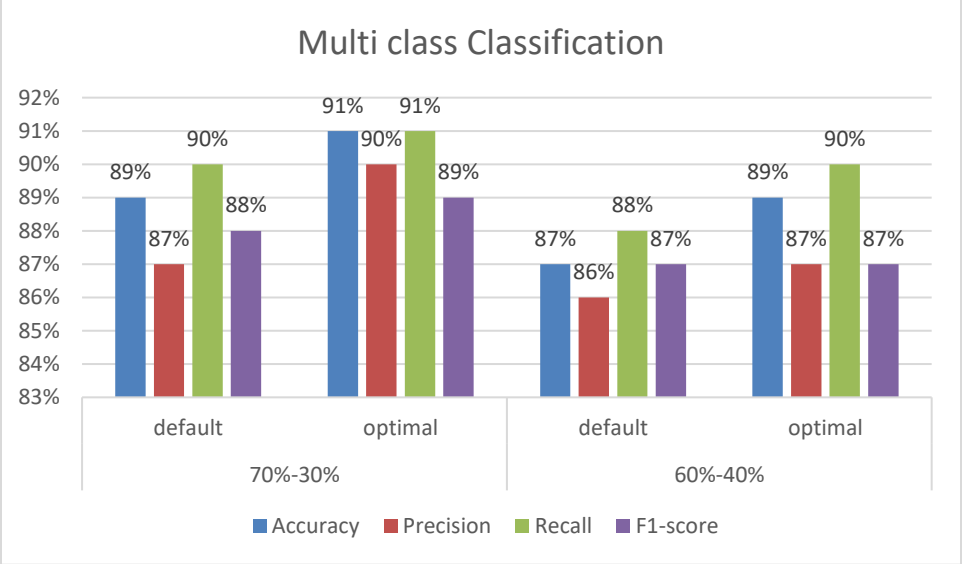


b)

Gambar 4. 13 Grafik batang masing-masing evaluasi pada klasifikasi *Binary*, a) grafik evaluasi pembagian data *train-test* 90%-10% dan 80%-20%, b), grafik evaluasi pembagian data *train-test* 70%-30% dan 60%-40%,



a)



b)

Gambar 4. 14 Grafik batang masing-masing evaluasi pada klasifikasi *multiclass*, a) grafik evaluasi pembagian data *train-test* 90%-10% dan 80%-20%, b) grafik evaluasi pembagian data *train-test* 70%-30% dan 60%-40%,

Dari gambar 4.12 dan 4.13 dapat dilihat masing-masing metrik evaluasi mengalami peningkatan yang signifikan sebesar 1% hingga 2% setelah dilakukan *tuning hyperparameter*. Pada grafik *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%. Secara lengkap hasil evaluasi klasifikasi dilihat pada lampiran 3.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan percobaan yang telah dilakukan dapat ditarik kesimpulan sebagai berikut:

1. Rancangan metode klasifikasi curah hujan dengan menggunakan model *XGBoost* telah berhasil dilakukan dengan metrik evaluasi yang cukup bagus, yaitu di atas 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 data set yang dilatih dalam membuat pohon keputusan, sehingga pada proporsi data set 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 meningkatkan akurasi model dalam melakukan prediksi, karena model memiliki pemahaman lebih baik dengan data yang banyak dibandingkan model dengan data latih yang sedikit.

5.2 Saran

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

1. Data set yang *imbalance* pada model dapat diatasi dengan metode *undersampling*, *oversampling*, atau metode penanganan data *imbalance* lainnya. Hal ini dapat memperbagus model dalam memahami data set.
2. Data set 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

Bansal, N., Singh, D., & Kumar, M. (2023). Computation of energy across the type-C piano key weir using gene expression programming and extreme gradient boosting (*XGBoost*) algorithm. *Energy Reports*, *9*, 310–321. https://doi.org/10.1016/j.egyr.2023.04.003

Canayaz, M. (2021). C+EffxNet: A novel hybrid approach for COVID-19 diagnosis on CT images based on CBAM and EfficientNet. *Chaos, Solitons and Fractals*, *151*. https://doi.org/10.1016/j.chaos.2021.111310

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

Das, Sibanjan., & Mert Cakmak, Umit. (2018). *Hands-On Automated Machine Learning : a beginner's guide to building automated machine learning systems using AutoML and Python.* Packt Publishing.

Deng, X., Liu, Q., Deng, Y., & Mahadevan, S. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, *340–341*, 250–261. https://doi.org/10.1016/j.ins.2016.01.033

Depto, D. S., Rizvee, M. M., Rahman, A., Zunair, H., Rahman, M. S., & Mahdy, M. R. C. (2023). Quantifying imbalanced classification methods for leukemia detection. *Computers in Biology and Medicine*, *152*. https://doi.org/10.1016/j.compbiomed.2022.106372

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

Shahani, N. M., Kamran, M., Zheng, X., Liu, C., & Guo, X. (2021). Application of gradient boosting machine learning algorithms to predict uniaxial compressive strength of soft sedimentary rocks at Thar coalfield. *Advances in Civil Engineering*, *2021*. https://doi.org/10.1155/2021/2565488

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*

```
## 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()


""""#### Univariate analysis

Data kategori
"""

target = ['classification']
feature = target[0]
count = data_baru[feature].value_counts()
percent = 100*data_baru[feature].value_counts(normalize=True)
df = pd.DataFrame({'jumlah sampel':count, 'persentase':percent.round(1)})
df

#menampilkan grafik
plt.figure(figsize=(4,3))
count.plot(kind='bar', title=feature)


"""Data Numerik"""
data_baru.hist(bins=50, figsize=(10,10))
plt.show()

""""#### Multivariate analysis

Data kategori
"""

fitur_kategori =
data_baru.select_dtypes(include='float64').columns.to_list()

for col in fitur_kategori:
    sns.catplot(x=col, y='classification', kind='bar', dodge=False,
height=3, aspect=2, data=data_baru, palette='Set3')
    plt.title("Rata-rata klasifikasi Relatif terhadap - {}".format(col))

"""Data Numerik"""
"""Matriks korelasi"""
```

```
plt.figure(figsize=(10,10))
sns.pairplot(data_baru, diag_kind='kde')

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)
pd.DataFrame(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"] + 0.01),
        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()
```

```
"""#### Univariate analysis

Data kategori
"""

target = ['classification']
feature = target[0]
count = data_baru[feature].value_counts()
percent = 100*data_baru[feature].value_counts(normalize=True)
df = pd.DataFrame({'jumlah sampel':count, 'persentase':percent.round(1)})
df

#menampilkan grafik
plt.figure(figsize=(4,3))
count.plot(kind='bar', title=feature)


"""Data Numerik"""
data_baru.hist(bins=50, figsize=(10,10))
plt.show()

"""#### Multivariate analysis

Data kategori
"""

fitur_kategori =
data_baru.select_dtypes(include='float64').columns.to_list()

for col in fitur_kategori:
    sns.catplot(x=col, y='classification', kind='bar', dodge=False,
height=3, aspect=2, data=data_baru, palette='Set3')
    plt.title("Rata-rata klasifikasi Relatif terhadap - {}".format(col))

"""Data Numerik"""
"""Matriks korelasi"""
plt.figure(figsize=(10,10))
sns.pairplot(data_baru, diag_kind='kde')

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)
pd.DataFrame(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"] + 0.01),
        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)
```

Lampiran 3. Hasil evaluasi klasifikasi

Klasifikasi biner 60-40 pembagian dataset

	Presisi	Recall	F1-score	Support
Hujan	0.28	0.17	0.21	78
Tidak Hujan	0.92	0.96	0.94	738
Micro Avg	0.88	0.88	0.88	816
Macro Avg	0.60	0.56	0.57	816
Weighted Avg	0.86	0.88	0.87	816
Samples Avg	0.88	0.88	0.88	816

Klasifikasi biner 70-30 pembagian dataset

	Presisi	Recall	F1-score	Support
Hujan	0.33	0.17	0.22	53
Tidak Hujan	0.92	0.96	0.95	559
Micro Avg	0.90	0.90	0.90	612
Macro Avg	0.63	0.57	0.59	612
Weighted Avg	0.87	0.90	0.88	612
Samples Avg	0.90	0.90	0.90	612

Klasifikasi biner 80-20 pembagian dataset

	Presisi	Recall	F1-score	Support
Hujan	0.41	0.28	0.33	32
Tidak Hujan	0.94	0.97	0.95	376
Micro Avg	0.91	0.91	0.91	408
Macro Avg	0.67	0.62	0.64	408
Weighted Avg	0.90	0.91	0.90	408
Samples Avg	0.91	0.91	0.91	408

Klasifikasi biner 90-10 pembagian dataset

	Presisi	Recall	F1-score	Support
Hujan	0.50	0.32	0.39	19
Tidak Hujan	0.93	0.97	0.95	185
Micro Avg	0.91	0.91	0.91	204
Macro Avg	0.72	0.64	0.67	204
Weighted Avg	0.89	0.91	0.90	204
Samples Avg	0.91	0.91	0.91	204

Klasifikasi *multiclass* 60-40 pembagian dataset

	Presisi	Recall	F1-score	Support
Ringan dan sedang	0.33	0.33	0.33	9
Sangat ringan	0.37	0.16	0.22	69
Tidak Hujan	0.92	0.96	0.94	738
Micro Avg	0.89	0.88	0.88	816
Macro Avg	0.54	0.48	0.50	816
Weighted Avg	0.86	0.88	0.87	816
Samples Avg	0.88	0.88	0.88	816

Klasifikasi *multiclass* 70-30 pembagian dataset

	Presisi	Recall	F1-score	Support
Ringan dan sedang	0.29	0.29	0.29	7
Sangat ringan	0.33	0.15	0.21	46
Tidak Hujan	0.92	0.97	0.95	559
Micro Avg	0.90	0.90	0.90	612
Macro Avg	0.51	0.47	0.48	612
Weighted Avg	0.87	0.90	0.88	612
Samples Avg	0.89	0.90	0.89	612

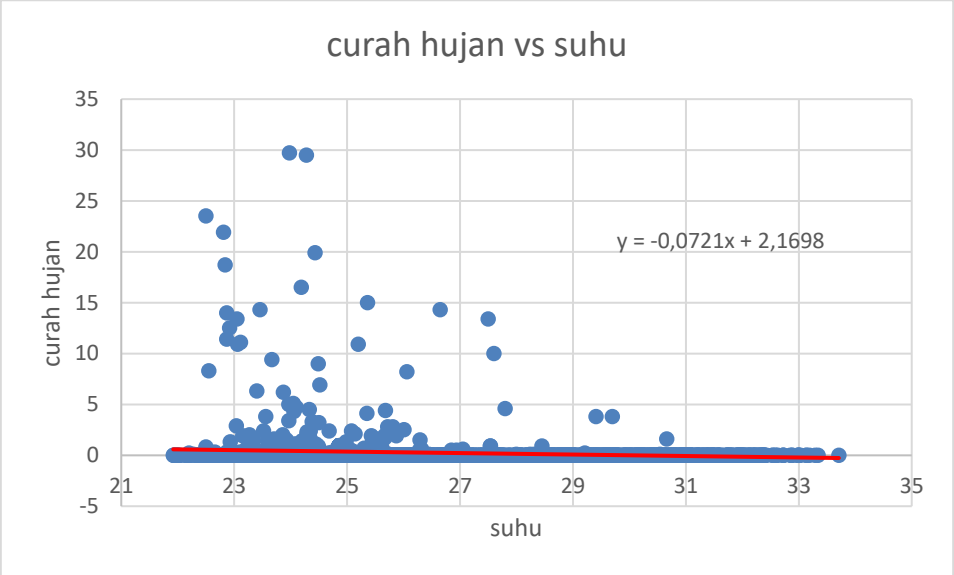
Klasifikasi *multiclass* 80-20 pembagian dataset

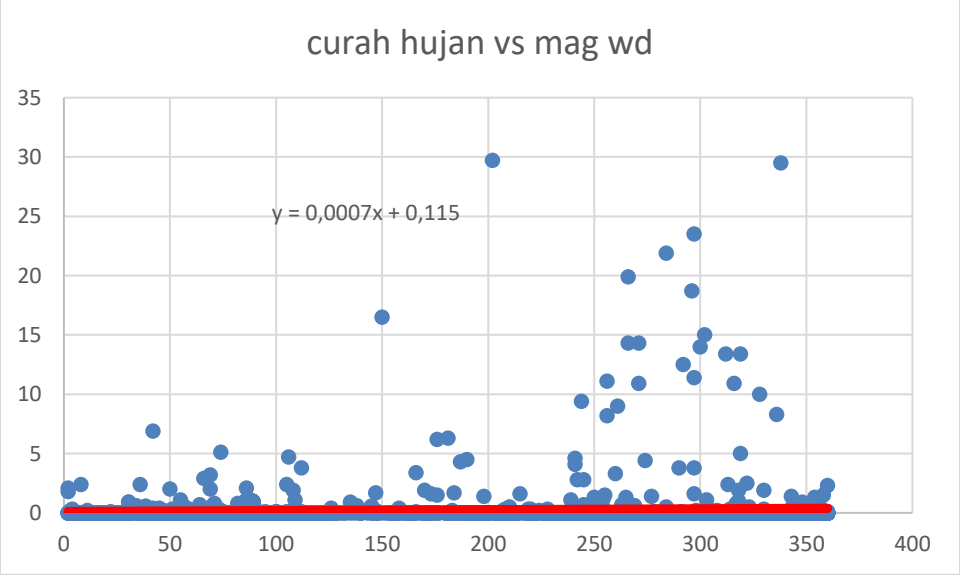
	Presisi	Recall	F1-score	Support
Ringan dan sedang	0.50	0.50	0.50	4
Sangat ringan	0.21	0.11	0.14	28
Tidak Hujan	0.94	0.97	0.95	376
Micro Avg	0.91	0.90	0.91	408
Macro Avg	0.55	0.52	0.53	408
Weighted Avg	0.89	0.90	0.89	408
Samples Avg	0.89	0.90	0.90	408

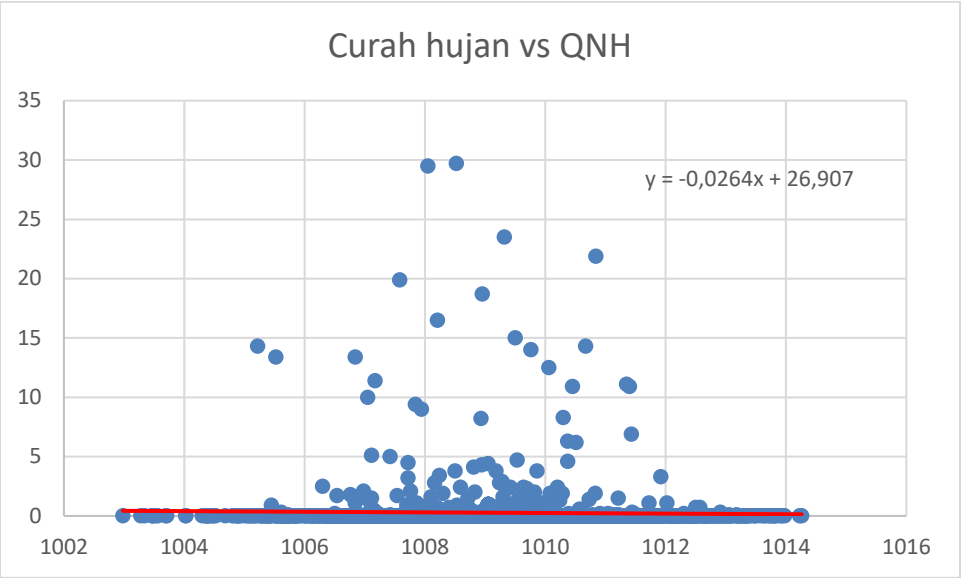
Klasifikasi *multiclass* 90-30 pembagian dataset

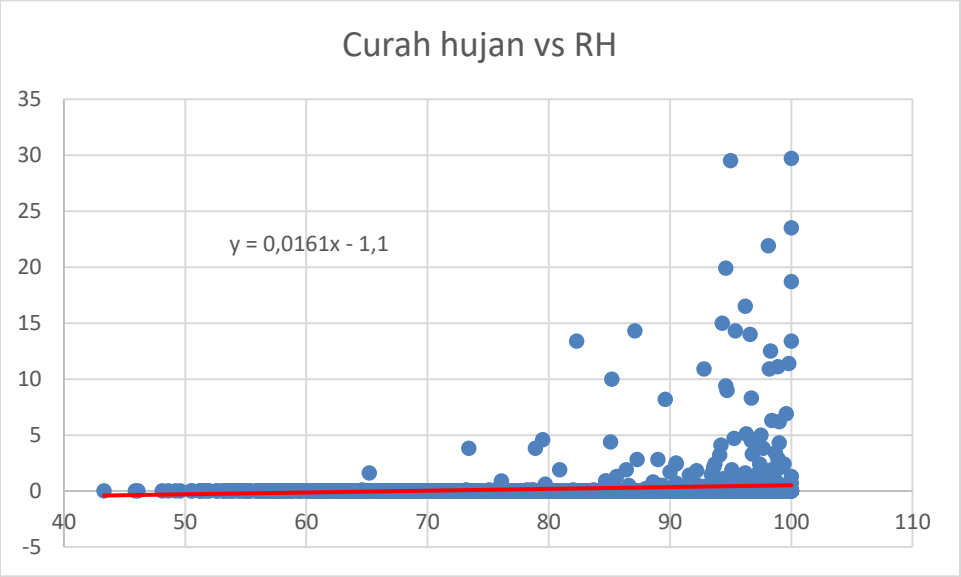
	Presisi	Recall	F1-score	Support
Ringan dan sedang	1.00	0.67	0.80	3
Sangat ringan	0.33	0.19	0.24	16
Tidak Hujan	0.93	0.97	0.95	185
Micro Avg	0.91	0.90	0.90	204
Macro Avg	0.76	0.61	0.66	204
Weighted Avg	0.89	0.90	0.89	204
Samples Avg	0.90	0.90	0.90	204

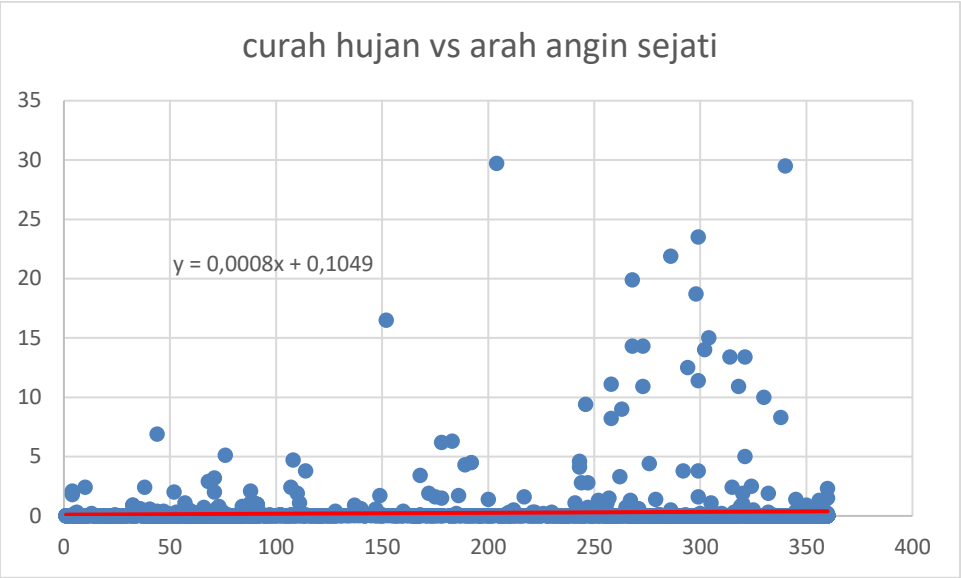
Lampiran 4. Grafik hubungan antara curah hujan dengan fitur independen

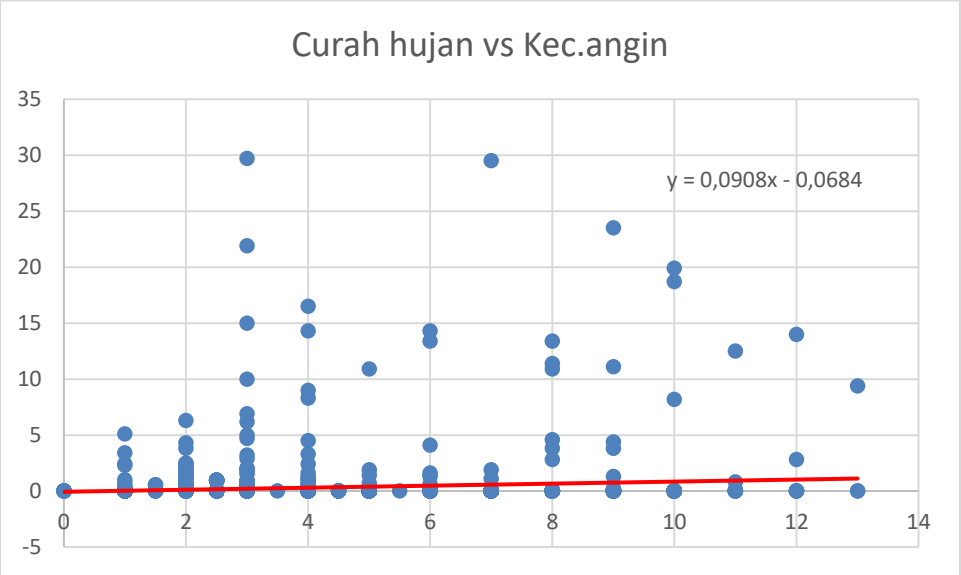





Curah hujan vs QNH
y = -0,0264x + 26,907
35
30
25
20
15
10
5
0
1002
1004
1006
1008
1010
1012
1014
1016


Curah hujan vs RH
y = 0,0161x - 1,1
35
30
25
20
15
10
5
0
-5
40
50
60
70
80
90
100
110


curah hujan vs arah angin sejati
y = 0,0008x + 0,1049
35
30
25
20
15
10
5
0
0
50
100
150
200
250
300
350
400


Curah hujan vs Kec.angin
y = 0,0908x - 0,0684
35
30
25
20
15
10
5
0
-5
0
2
4
6
8
10
12
14