Indones. J. Math. Nat. Sci. 47(1) 2024
Indonesian Journal of Mathematics and Natural Sciences
https://journal.unnes.ac.id/journals/JM/index

Perbandingan Metode *Tree Based Classification* untuk Masalah Klasifikasi Data *Body Mass Index*

Rifdah Nur Alifah, Mohamad Khoirun Najib*, Sri Nurdiati, Annisa Permata Sari, Karen Herlambang, Noval, Dini Tri Putri Br Ginting, Syifa Noer Sya'adah

Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, IPB University, Jalan Meranti Wing 22 Level 5, Kampus IPB Dramaga, Bogor 16680, Indonesia.
E-mail: mkhoirun_najib@apps.ipb.ac.id

Diterima 30 Maret 2024 Disetujui 05 Mei 2024

Abstrak

Body mass index (BMI) atau indeks massa tubuh merupakan salah satu indikator yang dapat mengawasi dan menjelaskan status gizi seseorang. Penelitian ini bertujuan untuk mengklasifikasikan BMI berdasarkan gender, tinggi badan, dan berat badan dengan menggunakan metode *tree based classification* yang terdiri atas model *decision tree classifier, random forest classifier, gradient boosting classifier,* dan *XGBoost* menggunakan bahasa pemrograman *python.* Model *tree based classification* tersebut akan mengklasifikasikan BMI ke dalam 6 kelas indeks. Hasil penelitian menunjukkan model klasifikasi *XGBoost* memiliki akurasi terbaik setelah dilakukan *tuning hyperparameter* dengan nilai akurasi *data test* 83,7%. Performa model terbaik sebelum *tuning hyperparameter* dihasilkan model *random forest* dengan nilai *F1-score (macro)* untuk *data test* sebesar 88%. Sementara itu, performa model terbaik setelah *tuning hyperparameter* dihasilkan model *XGBoost* dengan nilai *F1-score (macro)* untuk *data test* dan *data train* masing-masing sebesar 79% dan 85%. Berdasarkan model *XGBoost*, variabel prediktor yang paling berkontribusi terhadap BMI adalah berat badan dengan nilai *permutation importance* 68,1%.

Kata kunci: *Confusion Matrix,* skor F1, pentingnya permutasi*, Tree Based Classification, True Positive Rate*

Abstract

Body mass index (BMI) is one of the indicators that can monitor and explain a person's nutritional status. This research aims to classify BMI based on gender, height, and weight using the Tree Based Classification method which consists of the decision tree classifier, random forest classifier, gradient boosting classifier, and XGBoost models using the python. The Tree Based classification model will classify BMI into 6 index classes. The results showed that the XGBoost classification model had the best accuracy after hyperparameter tuning with a test data accuracy value of 83.7%. The best model performance before hyperparameter tuning is produced by the Random Forest model with an F1-score (macro) value for test data of 88%. Meanwhile, the best model performance after hyperparameter tuning is produced by the XGBoost model with F1-score (macro) values for test data and train data of 79% and 85%, respectively. Based on the XGBoost model, the predictor variable that contributes most to BMI is body weight with a permutation importance value of 68.1%.

Key words: *Confusion Matrix, F1-Score, Permutation Importance, Tree Based Classification, True Positive Rate*

How to cite:
Alifah RN, Najib MK, Nurdiati S, Sari AP, Herlambang K, Noval, Ginting DTPB, Sya'adah SN. (2024). Perbandingan metode tree based classification untuk masalah klasifikasi data body mass index. *Indonesian Journal of Mathematics and Natural Sciences*, 47(1), 49-65.

PENDAHULUAN

Era teknologi yang sudah berkembang saat ini dapat memudahkan manusia dalam memenuhi kebutuhan dalam menjalankan aktivitasnya, salah satunya adalah dalam memenuhi gizi. Akan tetapi, banyak negara di dunia baik negara berkembang maupun negara maju masih memiliki masalah kesehatan di bidang gizi. Salah satu negara berkembang yang memiliki masalah gizi dan menghadapi beban ganda gizi adalah Indonesia. Ketidakseimbangan asupan dan kebutuhan nutrisi pada tubuh

dapat menyebabkan terjadinya masalah gizi. Apabila asupan gizi dan energi tidak seimbang maka dapat mengakibatkan peningkatan berat badan ataupun kekurangan gizi (Wulandari *et al*., 2023).

Saat ini, Indonesia sedang menghadapi masalah kurang gizi yang berakibat pada kekurusan tubuh serta kelebihan gizi yang berakibat pada kegemukan atau obesitas. Kurangnya berat badan atau kekurusan disebabkan oleh kurangnya asupan gizi dan energi dalam tubuh, sedangkan kegemukan atau obesitas disebabkan oleh kelebihannya asupan gizi daripada energi yang dikeluarkan oleh tubuh (Wulandari *et al.,* 2023). Menurut riset kesehatan dasar kementerian kesehatan, terdapat perbaikan dalam status gizi buruk di Indonesia yang pada tahun 2013 sebesar 37,6% (Riskesdas, 2018) menjadi 21,6 pada tahun 2022 (SSGI, 2022). Meskipun terjadi penurunan, namun belum mencapai tingkat signifikan karena batas prevalensi gizi buruk yang ditetapkan oleh organisasi kesehatan dunia (WHO) adalah 20%. Demikian pula, proporsi anak dengan status gizi kurang menurun dari 19,6% pada tahun 2013 menjadi 17,1% pada tahun 2022 (SSGI, 2022). Selain masalah gizi buruk dan gizi kurang, angka obesitas di Indonesia juga cukup tinggi. Berdasarkan data UNICEF (2022), prevalensi kelebihan berat badan dan obesitas mengalami peningkatan yang cukup besar dari tahun 2013 sampai 2018. Peningkatan tersebut dari dilihat dari 9,2% menjadi 20% pada anak-anak usia 5-12 tahun, dari 1,9% menjadi 14,8 % pada anak-anak usia 13-18 tahun, dan dari 21,7% menjadi 35,4 % pada orang dewasa. Setelah melihat masalah yang terjadi di Indonesia akibat kekurangan gizi dan kelebihan gizi, maka perlu dilakukan tindakan preventif. Salah satunya adalah dengan mengetahui klasifikasi indeks massa tubuh seseorang agar dapat diterapkan tindakan pencegahan terhadap masalah gizi yang ada.

Body mass index (BMI) atau indeks massa tubuh merupakan salah satu indikator sederhana yang dapat mengawasi dan menjelaskan status gizi orang dewasa, terutama dalam kategori kekurangan gizi dan kelebihan berat badan atau obesitas (Lukas *et al*., 2017). Menurut WHO (2010), BMI diklasifikasikan menjadi enam kategori seperti yang ditunjukkan pada Tabel 1.

Tabel 1. Definisi BMI dan klasifikasinya

Klasifikasi BMI	Definisi BMI (Kg/m^2)
Underweight	< 18,5
Normal	18,5 – 24,9
Kegemukan	25,0 – 29,9
Obesitas kelas 1	30,0 – 34,9
Obesitas kelas 2	35,0 – 39,9
Obesitas kelas 3	> 40

Metode yang digunakan untuk mengklasifikasikan BMI berdasarkan faktor gender, berat badan, dan tinggi badan dapat dimodelkan dengan menggunakan metode *tree based classification*. *tree based classification* terdiri atas empat metode dengan basis metodenya adalah *decision tree*. *Decision tree* merupakan jenis struktur berbasis pohon yang digunakan untuk memprediksi hasil numerik dari variabel dependen dengan proses pembelajaran dengan data baru yang dikelompokkan menurut *training samples* yang sudah ada (Rathore *et al.,* 2016). *Random forest* adalah evolusi *bagging*, yang bertujuan untuk mengurangi varians model statistik dan mensimulasikan variabilitas data dengan mengekstrak secara acak sampel *bootstrap* dari set pelatihan dan prediksi agregat untuk catatan baru. *Random forest* ini merupakan salah satu algoritma klasifikasi *ensemble* berbasis pohon yang banyak digunakan (Effendy *et al.,* 2022). *Gradient Boosting* adalah teknik *supervised learning* berbasis *decision tree*. Algoritma dimulai dari menghasilkan pohon klasifikasi awal dan terus menyesuaikan pohon baru melalui minimisasi fungsi kerugian (Natekin & Knoll, 2013). *XGBoost* merupakan sebuah algoritma pengembangan dari teknik *gradient tree boosting* yang menggunakan pendekatan algoritma *ensemble*. Algoritma ini efektif dalam menangani permasalahan *machine learning* yang memiliki skala besar. Keputusan untuk memilih metode *XGBoost* didasarkan pada keberadaan fitur tambahan yang membantu mempercepat perhitungan sistem dan mencegah terjadinya *overfitting* (Yulianti *et al*., 2022).

Penelitian ini bertujuan untuk mengklasifikasikan BMI berdasarkan gender, tinggi badan, dan berat badan dengan menggunakan metode *decision tree classifier, random forest classifier, gradient boosting classifier,* dan *XGBoost*. Model yang sudah terbentuk dari keempat metode tersebut akan dibandingkan dengan melihat keakuratan model dan performa berdasarkan *confusion matrix* dan dilakukan analisis kontribusi faktor untuk melihat faktor yang paling mempengaruhi BMI.

METODE

Dataset

Penelitian ini menggunakan data sekunder yang diperoleh dari halaman *web* Kaggle yang dapat diakses pada https://www.kaggle.com/datasets/sjagkoo7/bmi-body-mass-index. *Dataset* yang akan digunakan pada penelitian ini mencakup informasi tentang gender (jenis kelamin), tinggi badan (cm), berat badan (kg), dan *Body Mass Index* (BMI). Variabel yang digunakan pada penelitian ini disajikan pada Tabel 2.

Tabel 2. Deskripsi variabel penelitian

No	Variabel	Keterangan
Y	*Body Mass Index* (BMI)	Indeks BMI individu dengan kategori 0 (sangat kurus), 1 (kurus), 2 (normal), 3 (kegemukan), 4 (obesitas), dan 5 (sangat obesitas)
X_1	Tinggi badan (*Height*)	Tinggi badan individu dalam satuan cm
X_2	Berat badan (*Weight*)	Berat badan individu dalam satuan kg
X_3	Gender	Jenis kelamin individu dalam kategori 0 (wanita), 1 (pria)

Body Mass Index (BMI) atau indeks massa tubuh merupakan parameter yang ditetapkan oleh *world health organization* (WHO) yang biasanya digunakan untuk mengklasifikasikan bagi penderita obesitas atau berat badan pada orang dewasa yang memiliki satuan (kg/m^2). Perhitungan BMI dapat dilakukan dengan menggunakan persamaan berikut.

$$BMI = \frac{Berat\ Badan}{(Tinggi\ Badan)^2}$$

Berat badan merupakan salah satu parameter yang digunakan dalam mengukur tubuh individu dalam satuan kilogram (kg) yang dapat digunakan untuk mengetahui kondisi tubuh serta menganalisa indeks massa tubuh seseorang (Rahman *et al.,* 2017). Sementara itu, tinggi badan merupakan salah satu indikator pengukuran yang dihitung dari atas kepala hingga ujung kaki dalam satuan sentimeter (cm) yang dapat digunakan untuk mengukur status gizi ataupun indeks massa tubuh seseorang (Murbawani, 2012).

Decision Tree Classifier

Decision Tree adalah salah satu metode klasifikasi yang mengklasifikasikan data terlatih yang diberi label ke dalam pohon atau aturan. Setelah pohon atau aturan diturunkan dalam tahap pembelajaran untuk menguji keakuratan pengklasifikasi, *test data* diambil secara acak dari *train data*. Setelah verifikasi keakuratan, data yang tidak berlabel diklasifikasikan menggunakan pohon atau aturan yang diperoleh pada tahap pembelajaran. Struktur *decision tree* mirip dengan *tree with a root node, a left subtree, and right subtree*. Node daun pada pohon mewakili label kelas. Busur dari satu node ke node lainnya menunjukkan kondisi pada atribut. Pohon dapat dibangun sebagai:

- Pemilihan atribut sebagai node akar dilakukan berdasarkan pemisahan atribut.
- Keputusan mengenai node yang akan direpresentasikan sebagai node terminal atau melanjutkan pemisahan node.
- Penugasan node terminal ke suatu kelas.

Setelah pohon selesai dibangun, pohon tersebut dipangkas untuk memeriksa kesesuaian dan kebisingan. Terakhir, pohon tersebut adalah pohon yang dioptimalkan. Keuntungan dari pendekatan terstruktur pohon adalah mudah dipahami dan diinterpretasikan, menangani atribut kategorikal dan numerik, tahan terhadap *outlier*, dan nilai yang hilang (Lavanya & Rani, 2012).

Proses pemisahan dalam pembentukan *decision tree* digunakan untuk mengukur keragaman sebuah *dataset*. Setiap simpul membagi *dataset* menjadi dua atau lebih *subdataset* yang semakin homogen. Tingkat ketidakpastian pada dataset diukur dengan metode entropi. Variabel dengan nilai *information gain* terbesar digunakan sebagai partisi terbaik. Misalkan *dataset D* dipartisi menjadi beberapa bagian $(D_1, D_2, \ldots, D_k)$. Rumus untuk menghitung entropi dan *information gain* yang dipelopori oleh Claude Shannon.

$$Entropi\ (D) = -\sum_{i=1}^{k} -p_i(p_i), \tag{1}$$

$$IG = Entropi\ (D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times Entropi(D_i), \tag{2}$$

dengan D adalah gugus data, k adalah jumlah partisi D, p_i adalah proporsi dari D_i terhadap D, IG adalah *information gain*, $|D_i|$ adalah jumlah observasi pada partisi ke-i, dan $|D|$ adalah jumlah observasi pada D.

Gini merupakan ukuran lain yang dapat digunakan untuk mengukur tingkat heterogenitas *dataset.* Proses partisi memperhatikan nilai $gini$ untuk menghitung nilai $gini\ decrease$. Perhitungan rumus dilakukan sebagai berikut:

$$Gini(D) = 1 - \sum_{i=1}^{k} (p_i^2), \tag{3}$$

$$GD = Gini(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times Gini(D_i), \tag{4}$$

dengan D adalah gugus data, k adalah jumlah partisi D, p_i adalah proporsi dari D_i terhadap D, GD adalah *information gain*, $|D_i|$ adalah jumlah observasi pada partisi ke-i, dan $|D|$ adalah jumlah observasi pada D.

(Ramadani, 2023)

Langkah-langkah pelatihan klasifikasi *decision tree* sesuai untuk algoritme pertumbuhan pohon rekursif yang elegan yang ditunjukkan sebagai berikut:

1. Diberikan kumpulan data berlabel Z, kriteria evaluasi fitur, dan kriteria pembuatan daun.
2. Mulai dengan pohon kosong dan pertimbangkan simpul akar.
3. Jika kriteria pembuatan daun terpenuhi, buatlah daun dan tetapkan padanya label yang paling mewakili data yang sampai pada simpul tersebut.
4. Jika tidak, tambahkan node perantara ke pohon. Gunakan kriteria evaluasi fitur untuk menemukan fitur terbaik dan ambang batas masing-masing. Gunakan ini untuk membagi data dan mengirim bagian-bagiannya ke node anak kiri dan kanan.
5. Ulangi dari langkah 3 untuk kedua node anak hingga seluruh Z didistribusikan ke dalam node daun.
6. Kembalikan pohonnya.

Random Forest Classifier

Random forest menghasilkan kumpulan *decision tree*. Untuk mencapai keberagaman di antara pohon keputusan dasar, Breiman memilih pendekatan pengacakan yang bekerja dengan baik dengan metode *bagging* atau subruang acak. Untuk menghasilkan setiap pohon di *random forest*, Breiman mengikuti langkah-langkah berikut: Jika jumlah catatan dalam set pelatihan adalah N, maka N catatan adalah diambil sampelnya secara acak tetapi dengan penggantian, dari data asli, ini sampel *bootstrap*. Sampel ini akan menjadi set pelatihan untuk menumbuhkan pohon. Jika ada masukan M variabel, sejumlah m < M dipilih sedemikian rupa sehingga pada masing-masing variabel simpul, m variabel dipilih secara acak dari M dan pemisahan terbaik pada atribut m ini digunakan untuk membagi node. Nilai m dijaga konstan selama pertumbuhan hutan. Setiap pohon ditanam semaksimal mungkin. Tidak ada pemangkasan.

Banyak pohon ditanam di hutan merupakan jumlah pohon ditentukan sebelumnya oleh parameter *N-tree*. Banyaknya variabel (m) yang dipilih pada setiap node juga disebut sebagai *mtry* atau k. Kedalaman pohon dapat dikontrol oleh parameter ukuran node (misalnya jumlah *instance* di node daun) yang biasanya dikendalikan untuk satu.

Setelah hutan dilatih atau dibangun, maka mengklasifikasikan *instance* baru, maka dijalankan di semua pohon yang tumbuh di dalam hutan. Setiap pohon memberikan klasifikasi untuk yang baru kejadian yang dicatat sebagai pemungutan suara. Suara dari semua pohon digabungkan dan kelasnya maksimal suara yang dihitung (suara terbanyak) dinyatakan sebagai klasifikasi contoh baru.

Dalam proses pembangunan hutan, saat sampel *bootstrap* ditetapkan diambil dengan pengambilan sampel dengan penggantian setiap pohon, sekitar $1/3$ dari contoh asli ditinggalkan. Kumpulan ini contoh disebut data OOB (*Out-of-bag*). Setiap pohon punya kumpulan data OOB miliknya sendiri yang digunakan untuk estimasi kesalahan pohon individu di hutan, disebut sebagai kesalahan OOB perkiraan. Algoritma *random forest* juga sudah ada di dalamnya fasilitas untuk menghitung kepentingan dan kedekatan variabel. Kedekatan digunakan untuk menggantikan nilai yang hilang dan *outlier* (Kulkarni & Sinha, 2013).

Klasifikasi *random forest* menggunakan metode *bagging* dan *random feature selection*. *Random forest* mengembangkan metode *bagging* dengan memilih acak peubah penjelas untuk mengurangi korelasi antar pohon yang terbentuk. Tahapan klasifikasi *random forest* pada gugus data yang terdiri atas n amatan dan p peubah penjelas (Sartono & Syafitri, 2010) sebagai berikut:

1. Tahapan *bootstrap*. *Bootstrap* adalah pengambilan contoh yang disertai dengan pengembalian. Pada tahap ini, n contoh acak diambil dari *dataset* latih.
2. Tahapan *random feature selection*. *Decision tree* disusun berdasarkan hasil *bootstrap* sebelumnya. Untuk setiap proses pemisahan, pilih m peubah penjelas secara acak dengan $m<p$. Selanjutnya, pemisahan terbaik dilakukan.
3. Langkah 1 dan 2 diulangi sebanyak k kali sehingga diperoleh *decision tree* sebanyak k.
4. Penggabungan (*agregasi*) hasil prediksi *decision tree* dilakukan sebanyak k dan *majority vote* (mengambil *vote* terbanyak) digunakan untuk menentukan hasil prediksi akhir.

Dalam masalah klasifikasi biner, metode *majority vote* dapat dilakukan sebagai berikut:

$$\hat{y} = \begin{cases} 1, & \sum_{i=0}^{k} y_i \geq \theta, \\ 2, & \sum_{i=0}^{k} y_i < \theta, \end{cases} \tag{5}$$

dengan $\hat{y}$ merupakan hasil prediksi, y_i merupakan hasil *decision tree* ke-i, dan θ merupakan ambang batas *majority vote* dengan *default* 0,5 (Ramadani, 2023).

Gradient Boosting Classifier

Gradient boosting adalah teknik *supervised learning* berbasis *decision tree*. *Gradient boosting* memerlukan konfigurasi *hyperparameter* pada tahap awal, kombinasi *hyperparameter* yang optimal mempengaruhi hasil prediksi *Gradient Boosting* (Suryana *et al.,* 2021). *Gradient Boosting* adalah konsep yang sangat penting karena menjadi dasar algoritma pembelajaran mesin populer seperti *XGBoost*. *Gradient boosting* menyesuaikan pohon keputusan secara berulang menggunakan kesalahan prediksi. Namun, pohon peningkat gradien biasanya lebih dalam daripada tunggul pohon keputusan dan biasanya memiliki kedalaman maksimum 3 hingga 6 (atau jumlah maksimum 8 hingga 64 simpul daun). *Gradient boosting* tidak menggunakan kesalahan prediksi untuk menetapkan bobot sampel. *Gradient boosting* digunakan secara langsung untuk membentuk variabel target untuk pemasangan pohon berikutnya. *Gradient boosting* menggunakan kecepatan pembelajaran *global* yang sama untuk setiap pohon.

Paper *Gradient Boosting* pertama kali ditulis oleh Friedman pada tahun 2001, *Greedy function approximation: a gradient boosting machine*. *Gradient boosting* membangun serangkaian pohon, di mana setiap pohon cocok dengan kesalahan, perbedaan antara label dan nilai prediksi dari pohon sebelumnya. Di setiap putaran, *ensemble* pohon meningkat seiring kami mendorong setiap pohon ke arah yang lebih benar melalui pembaruan kecil. Pembaruan ini didasarkan pada gradien kerugian, yang merupakan asal mula peningkatan gradien mendapatkan namanya.

Langkah-langkah berikut memperkenalkan algoritma *gradient boosting* secara umum.

1. Inisialisasi model untuk mengembalikan nilai prediksi yang konstan. Untuk ini, gunakan simpul akar pohon keputusan, yaitu pohon keputusan dengan satu simpul daun. Nyatakan nilai yang dikembalikan oleh pohon sebagai $\hat{y}$, dan cari nilai ini dengan meminimalkan fungsi kerugian terdiferensiasi L yang didefinisikan sebagai berikut:
$$F_0(x) = arg\min_{\hat{y}} \sum_{i=1}^{n} L(y_i, \hat{y}) \tag{6}$$
dengan n mengacu pada n contoh pelatihan.
2. Untuk setiap pohon $m = 1,2,..,M$, dengan M adalah jumlah total pohon yang ditentukan pengguna, lakukan perhitungan berikut yang dijelaskan pada langkah 2a hingga 2d di bawah ini:
    a. Hitung selisih antara nilai prediksi $F(x_i) = \hat{y}_i$ dan label kelas y_i. Nilai ini terkadang disebut respons semu atau sisa semu. Secara lebih formal, sisa semu dapat ditulis sebagai gradien negatif dari fungsi kerugian terhadap nilai prediksi:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad for\ i = 1, \ldots, n \tag{7}$$

Perhatikan bahwa pada notasi di atas $F(x)$ merupakan prediksi pohon sebelumnya, $F_{m-1}(x)$. Jadi, pada putaran pertama, ini mengacu pada nilai konstanta dari pohon (simpul daun tunggal) dari langkah 1.

b. Pasangkan pohon ke sisa semu r_{im}. Gunakan notasi R_{jm} untuk menunjukkan $j = 1 \ldots J_m$ simpul daun dari pohon yang dihasilkan dalam iterasi m.

c. Untuk setiap simpul daun R_{jm}, hitung nilai keluaran berikut

$$\gamma_{jm} = arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma) \tag{8}$$

γ_{jm} dihitung dengan meminimalkan fungsi kerugian. Pada titik ini, node daun R_{jm} mungkin berisi lebih dari satu contoh pelatihan, oleh karena itu dilakukan penjumlahan.

d. Perbarui model dengan menambahkan nilai keluaran γ_m ke pohon sebelumnya:

$$F_m(x) = F_{m-1}(x) + \eta\gamma_m \tag{9}$$

Namun, alih-alih menambahkan nilai prediksi penuh dari pohon saat ini γ_m ke pohon sebelumnya F_{m-1}, skalakan γ_m dengan kecepatan pembelajaran η, yang biasanya merupakan nilai kecil antara 0,01 dan 1. Dengan kata lain, memperbarui model secara bertahap dengan mengambil langkah-langkah kecil, yang membantu menghindari *overfitting*.

Gradient boosting classifier menggunakan fungsi kerugian logistik. Untuk contoh pelatihan tunggal, kerugian logistik dapat ditentukan sebagai berikut:

$$L_i = -y_i\ log\ p_i + (1 - y_i) log(1 - p_i) \tag{10}$$

$$\hat{y} = log(odds) = log(\frac{p}{1-p}) \tag{11}$$

Fungsi $log(odds)$ digunakan untuk menulis ulang fungsi logistik sebagai berikut (menghilangkan langkah perantara):

$$L_i = log(1 + e^{\widehat{y_i}}) - y_i\widehat{y_i} \tag{12}$$

Definisikan turunan parsial dari fungsi kerugian terhadap $log(odds), \hat{y}$. Turunan dari fungsi kerugian ini terhadap $log(odds)$ adalah:

$$\frac{\partial L_i}{\partial \hat{y}_i} = \frac{e^{\hat{y}_i}}{1 + e^{\hat{y}_i}} - y_i = p_i - y_i \tag{13}$$

Setelah menentukan aspek matematika, langkah peningkatan gradien umum 1 hingga 2d dari bagian sebelumnya ditinjau dan diformulasikan kembali menjadi

1. Buat node *root* yang meminimalkan kehilangan logistik. Ternyata kerugiannya bisa diminimalkan jika simpul akar mengembalikan $log(odds), \hat{y}$.
2. Untuk setiap pohon $m = 1, .., M$, dengan M adalah jumlah total pohon yang ditentukan pengguna, lakukan penghitungan berikut yang dijelaskan pada langkah 2a hingga 2d:
   a. Konversi $log(odds)$ menjadi probabilitas menggunakan fungsi logistik yang biasa digunakan dalam regresi logistik

   $$p = \frac{1}{1 + e^{-\hat{y}}} \tag{14}$$

   Kemudian, hitung sisa semu, yang merupakan turunan parsial negatif dari kerugian terhadap $log(odds)$, yang merupakan selisih antara label kelas dan probabilitas yang diprediksi:

   $$-\frac{\partial L_i}{\partial \hat{y}_i} = y_i - p_i \tag{15}$$

   b. Pasangkan pohon baru ke sisa semu.
   c. Untuk setiap simpul daun R_{jm}, hitung nilai γ_{jm} yang meminimalkan fungsi kerugian logistik. Ini mencakup langkah ringkasan untuk menangani node daun yang berisi beberapa contoh pelatihan:

   $$\gamma_{jm} = arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma) = \log(1 + e^{y_i + \gamma}) - y_i(\widehat{y_i} + \gamma) \tag{16}$$

   Hasilnya adalah sebagai berikut:

$$\gamma_{jm} = \frac{\sum_i y_i - p_i}{\sum_i p_i(i - p_i)} \quad (17)$$

Perhatikan bahwa penjumlahan di sini hanya pada contoh pada node yang sesuai dengan node daun R_{jm}, dan bukan set pelatihan lengkap.

d. Perbarui model dengan menambahkan nilai gamma dari langkah 2c dengan laju η:

$$F_m(x) = F_{m-1}(x) + \eta\gamma_m \quad (18)$$

(Raschka *et al.*, 2022)

XGBoost Classifier

Salah satu teknik dalam *machine learning* untuk klasifikasi berdasarkan *gradient boosting decision tree* adalah *extreme gradient boosting* (*XGBoost*). *XGBoost* dikenalkan pertama kali oleh Friedman pada tahun 2001. Dalam penelitiannya, *boosting* dan optimasi dalam membangun *gradient boosting machine* dihubungkan oleh Friedman. Metode *XGBoost* merupakan sebuah pengembangan dari *gradient tree boosting* yang menggunakan algoritma *ensemble.* Algoritma *XGBoost* ini efektif dalam menangani kasus *machine learning* berskala besar. Metode *XGBoost* dipilih sebab adanya fitur tambahan yang berguna untuk mempercepat sistem perhitungan dan mengurangi terjadinya *overfitting* (Yulianti *et al*., 2022). Perbedaan antara *gradient boosting* dan *XGBoost* terletak pada cara penambahan "*weak learner*", yaitu pada *XGBoost*, proses penambahan "*weak learner*" tidak berlangsung secara berturut-turut, melainkan dilakukan secara *multi-threaded* (Abdurrahman *et al.,* 2022).

Nilai prediksi pada langkah t adala $\hat{y}_i^{(t)}$dengan

$$\hat{y}_i^{(t)} = \sum_{k=1}^{n} f_k(x_i), \quad (19)$$

$f_k(x_i)$ menggambarkan model pohon. Untuk y_i diperoleh dari perhitungan berikut:

$$\begin{aligned} \hat{y}_i^{(0)} &= 0 \\ \hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ &\vdots \\ \hat{y}_i^{(t)} &= \hat{y}_i^{(t-1)} + f_t(x_i) \\ \hat{y}_i^{(t)} &= \sum_{k=1}^{n} f_k(x_i), \end{aligned} \quad (20)$$

dengan $\hat{y}_i^{(t)}$ adalah model akhir pohon, $\hat{y}_i^{(t-1)}$ adalah model pohon yang dihasilkan sebelumnya, $f_t(x_i)$ adalah model baru yang dibangun, t adalah jumlah total model dari *base tree*.

Titik data individu digunakan untuk menyusun fungsi objektif. Penjumlahan nilai objektif dari setiap titik data adalah nilai objektif dataset. Misalkan £ merupakan nilai objektif dataset dan $l(y_i, \hat{y}_i)$ merupakan titik data ke-i. Hal ini bergantung pada label sebenarnya (y_i) dan label prediksi $(\hat{y}_i)$. Persamaan yang menyatakan hubungan antara nilai objektif *dataset* dan titik data adalah sebagai berikut:

$$£ = \sum_{i=1}^{t} l(y_i, \hat{y}_i) \quad (21)$$

Pemilihan titik data bergantung pada masalah prediksi. Pilihan yang paling umum untuk klasifikasi biner didefinisikan sebagai berikut:

$$l(y_i, \hat{y}_i) = -y\ log(sigmoid(\hat{y})) + (y - 1)log(1 - sigmoid(\hat{y})) \quad (22)$$

Hal penting dalam algoritme *XGBoost* adalah penentuan jumlah pohon dan kedalaman pohon (*depth*). Algoritme yang optimal dapat dilakukan dengan mencari klasifikasi baru yang dapat mengurangi *loss function* dengan fungsi objektif sebagai berikut:

$$Obj^{(t)} = \sum_{i=1}^{t} l(y_i, \hat{y}_i) + \sum_{i=1}^{t} \Omega(f_i) \quad (23)$$

dengan $\hat{y}_i$ adalah nilai prediksi, y_i adalah nilai aktual, $l(y_i, \hat{y}_i)$ adalah *loss function*, dan $\Omega(f_i)$ adalah regularisasi. Karena model merupakan fungsi parameter, model dilatih secara aditif menggunakan $\hat{y}_i^{(t)}$ pada prediksi ke-i dan iterasi ke-t. f_t ditambahkan untuk meminimalkan *loss function* sehingga didapatkan persamaan sebagai berikut:

$$Obj^{(t)} = \sum_{i=1}^{t} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_i) + constant \tag{24}$$

Selanjutnya mengubah target akhir dari *loss function* dan melatih sesuai target *loss function* sebagai berikut:

$$\begin{gathered} Obj^{(t)} = \sum_{i=1}^{t} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i))] + \Omega(f_i) \\ g_i: \partial_{y_i(t-1)} l\left(y_i, \widehat{y_i}^{(t-1)}\right), \\ h_i: \partial^2{}_{y_i(t-1)} l\left(y_i, \widehat{y_i}^{(t-1)}\right). \end{gathered} \tag{25}$$

dengan g_i dan h_i adalah urutan pertama dan kedua *gradient* pada *loss function*. Regulasisasi $\Omega(f_i)$ digunakan untuk mengurangi kompleksitas model dan meningkatkan kegunaan pada data lainnya, dapat dihitung menggunakan persamaan sebagai berikut:

$$\Omega(f_i) = \gamma T_i + \frac{1}{2}\lambda ||\omega_i||^2 \tag{26}$$

dengan T_i merupakan jumlah mode *leaf* pada pohon ke-i, ω_i merupakan bobot *leaf* pada pohon ke-i, dan λ, γ merupakan koefisien dengan nilai default untuk $\lambda = 1$.

Ukuran Performa Model

Setelah melatih model, evaluasi performa model tersebut dinilai menggunakan *confusion matrix* berdasarkan *test data* dan *train data* saat sebelum dan sesudah *tuning hyperparameter*. Pada kasus klasifikasi enam kelas, *confusion matrix* yang diperoleh dapat dilihat pada Gambar 1. Nilai $C_{0,0}$ menunjukkan banyaknya data yang sebenarnya kelas 0 diprediksi oleh model pada kelas 0 juga, sedangkan $C_{0,1}$ menunjukkan data yang sebenarnya adalah kelas 0, tetapi diprediksi kelas 1 oleh model. Interpretasi yang sama berlaku nilai yang lain pada Gambar 1.

		Prediksi					
		0	1	2	3	4	5
Aktual	0	$C_{0,0}$	$C_{0,1}$	$C_{0,2}$	$C_{0,3}$	$C_{0,4}$	$C_{0,5}$
	1	$C_{1,0}$	$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	$C_{1,4}$	$C_{1,5}$
	2	$C_{2,0}$	$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	$C_{2,4}$	$C_{2,5}$
	3	$C_{3,0}$	$C_{3,1}$	$C_{3,2}$	$C_{3,3}$	$C_{3,4}$	$C_{3,5}$
	4	$C_{4,0}$	$C_{4,1}$	$C_{4,2}$	$C_{4,3}$	$C_{4,4}$	$C_{4,5}$
	5	$C_{5,0}$	$C_{5,1}$	$C_{5,2}$	$C_{5,3}$	$C_{5,4}$	$C_{5,5}$

Gambar 1. *Confusion matrix* pada klasifikasi 6 kelas berbeda

Berdasarkan Gambar 1 *confusion matrix* yang dihasilkan pada klasifikasi 6 kelas berbeda, nilai akurasi model prediksi dapat dihitung. Akurasi dapat ditentukan menggunakan persamaan dengan kasus klasifikasi n kelas sebagai berikut:

$$Accuracy = \frac{\sum_{i=0}^{n} C_{i,i}}{\sum_{j=0}^{n} \sum_{i=0}^{n} C_{i,j}} \tag{27}$$

Nilai akurasi yang telah ditentukan merupakan perbandingan antara jumlah diagonal utama (*trace*) dan jumlah semua titik pada *confusion matrix*. Selain akurasi, terdapat ukuran lain yang dapat ditentukan melalui *confusion matrix*, yaitu nilai *true positive rate* (*recall*), nilai *positive predictive value* (presisi), dan nilai *F1-score*, masing-masing untuk kelas i dengan persamaan sebagai berikut:

$$TPR(C_i) = \frac{C_{i,i}}{\sum_{j=0}^{n} C_{i,j}} \tag{28}$$

$$PPV(C_i) = \frac{C_{i,i}}{\sum_{j=0}^{n} C_{j,i}} \tag{29}$$

$$F_1(C_i) = 2 \cdot \frac{TPR(C_i) \cdot PPV(C_i)}{TPR(C_i) + PPV(C_i)} \tag{30}$$

dengan $i = 1, 2, 3, \ldots, n$. Berdasarkan persamaan tersebut, setiap kelas masing-masing akan memiliki nilai *recall*, presisi, dan *F1-score*, sehingga dalam penilaian performa model dapat dilakukan secara lebih umum untuk setiap nilai *recall,* presisi, dan *F1-score* menggunakan pendekatan makro, yaitu:

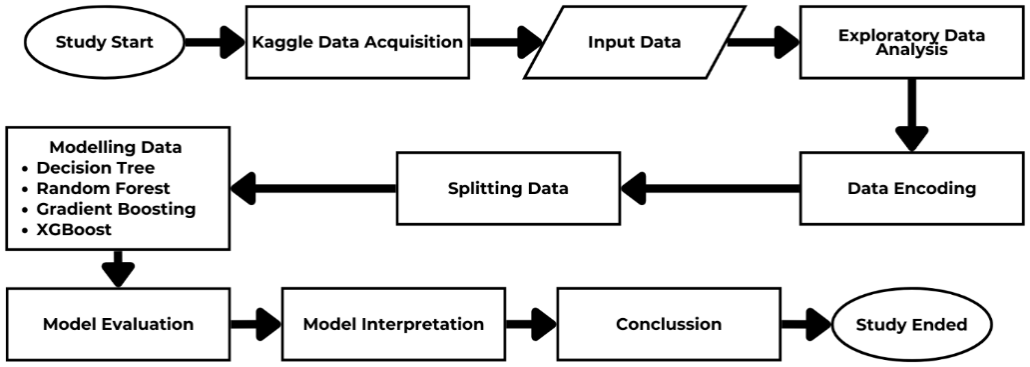
$$TPR(makro) = \frac{1}{n}\sum_{i=0}^{n} TPR(C_i) \tag{31}$$

$$PPV(makro) = \frac{1}{n}\sum_{i=0}^{n} PPV(C_i) \tag{32}$$

$$F_1(makro) = 2 \cdot \frac{TPR(makro) \cdot PPV(makro)}{TPR(makro) + PPV(makro)} \tag{33}$$

Adapun $TPR(makro)$, $PPV(makro)$, dan $F_1(makro)$ masing-masing merupakan ukuran yang digunakan untuk menilai *recall,* presisi, dan *F1-score* model prediksi dengan pendekatan makro.

Alur Metode Penelitian



Gambar 2. Diagram metode penelitian

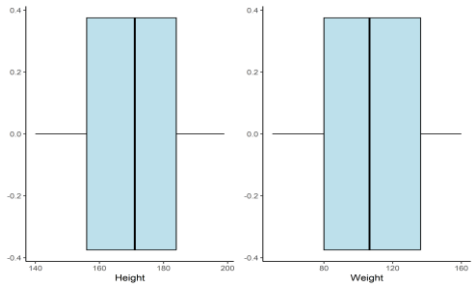
HASIL DAN PEMBAHASAN

Eksplorasi Data dan Praproses Data

Eksplorasi data merupakan upaya untuk memahami dan mendapatkan informasi dari suatu *dataset*. Namun, sebelum melakukan eksplorasi data perlu dilakukan *data preparation* untuk memastikan bahwa data siap untuk dijalankan melalui proses analisis atau pemodelan data. Pada penelitian kali ini, *dataset* berisi 400 data dengan variabel prediktor yang digunakan adalah gender, tinggi badan (*height*), berat badan (*weight*), sedangkan variabel responnya adalah *Body Mass Index* (BMI).

Data preparation dilakukan dengan penghapusan data duplikat dan *variable encoding*. Penghapusan data duplikat dilakukan untuk menghindari bias dalam analisis, meningkatkan efisiensi analisis, dan meningkatkan ketelitian model. Jumlah *dataset* yang awalnya adalah 400, setelah penghapusan data duplikat, menjadi 392. Selanjutnya, dilakukan *variable encoding* untuk mengubah variabel kategorik menjadi variabel biner sehingga dapat diolah oleh algoritma dalam analisis data (Kunanbayev *et al.,* 2021). Dalam penelitian ini, dilakukan *encoding* untuk variabel gender, dengan perempuan diwakilkan angka 0 dan laki-laki diwakilkan angka 1.

Setelah *data preparation* selesai, dilakukan eksplorasi data menggunakan berbagai metode visualisasi. *Boxplot* untuk mendeskripsikan data dari variabel *weight* dan *height*. Selain itu, dilakukan juga analisis korelasi menggunakan *heatmap* korelasi untuk memvisualisasikan hubungan antara variabel prediktor dan variabel respons dalam suatu *dataset* (Leni *et al.*, 2023). Eksplorasi data juga melibatkan pembentukan diagram batang untuk memeriksa proporsi setiap kelas dalam BMI.

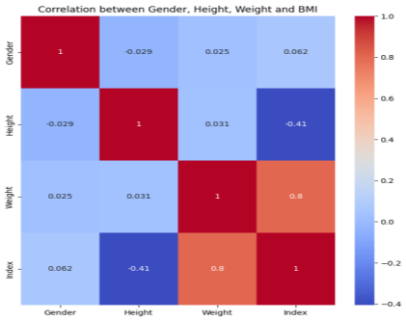


Gambar 3. *Boxplot* untuk tinggi bada*n (height)* dan berat badan *(weight)*

Box plot pada Gambar 3 untuk berat badan (*weight*) pada data BMI terdapat berat badan minimum dan maksimum, yaitu 50 dan 160 kg dengan rata-rata berat badan 106 kg. Sementara itu, tinggi badan (*height*) pada data BMI terdapat tinggi badan minimum dan maksimum, yaitu 140 dan 199 sentimeter dengan rata-rata berat badan 170 sentimeter. Berdasarkan sebaran *boxplot* tersebut bahwa tidak terdapat *outlier* atau pencilan pada data sehingga dapat dilanjutkan untuk melakukan analisis.

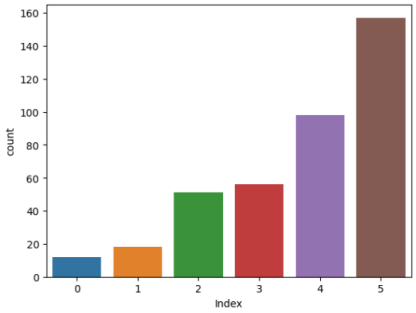
Selanjutnya, eksplorasi data menggunakan *heatmap* korelasi. Persamaan yang digunakan dalam *heatmap* korelasi adalah persamaan korelasi pearson. Persamaan ini digunakan untuk mengukur tingkat korelasi linear antara dua variabel dengan menghasilkan nilai koefisien korelasi (Leni *et al*., 2023). Rentang nilai koefisien korelasi berada antara -1 hingga 1. Nilai -1 menunjukkan hubungan negatif sempurna (terbalik), nilai 0 menunjukkan tidak ada hubungan, sementara nilai 1 menunjukkan hubungan positif sempurna (Wibowo & Kurniawan, 2020).

Gambar 4 mengindikasikan bahwa tinggi badan memiliki korelasi negatif dengan BMI, sementara gender dan berat badan memiliki korelasi positif dengan BMI. Korelasi tertinggi dengan BMI ditemukan pada berat badan, mencapai nilai 0,8. Sebaliknya, tinggi badan memiliki korelasi terendah dengan BMI, yakni -0,41. Sementara itu, gender menunjukkan korelasi sebesar 0,062 dengan BMI. Informasi tentang hubungan antar variabel prediktor juga dapat diperoleh dari gambar 4. Tidak ada nilai korelasi antar variabel prediktor yang melebihi 0,1 atau -0,1, menunjukkan bahwa tidak terdapat indikasi multikolinearitas. Multikolinearitas adalah kondisi saat terjadi korelasi yang besar antara variabel prediktor atau variabel prediktor tidak saling bebas (Sriningsih *et al.,* 2018).



Gambar 4. *Heatmap* nilai koefisien korelasi variabel prediktor dan variabel respon

Berdasarkan Gambar 5, dari proses eksplorasi data dapat dihasilkan diagram batang yang menunjukkan proporsi setiap kelas dalam BMI. Kelas 5 memiliki proporsi paling besar, sementara kelas 0 memiliki proporsi paling sedikit. Gambar 5 menunjukkan bahwa proporsi kelas 5 memiliki selisih yang signifikan dibandingkan dengan kelas 0, 1, 2, 3, dan 4. Hal ini mengindikasikan adanya ketidakseimbangan data atau *imbalance data* (Wijayanti *et al.,* 2021). Pada penelitian ini, tidak dilakukan *oversampling* sebagai upaya untuk mengatasi ketidakseimbangan data. Keputusan ini didasarkan pada pertimbangan bahwa *oversampling* dapat meningkatkan risiko *overfitting* selama analisis data. *Overfitting* terjadi ketika model terlalu khusus pada *train data*, memberikan kinerja optimal pada data tersebut tetapi gagal dalam generalisasi pada *test data.* Ini disebabkan oleh kesulitan model untuk menangani informasi pada *test data* yang berbeda, karena cenderung mengingat semua data, termasuk *noise* (Ying, 2018).



Gambar 5. Proporsi setiap kelas dalam BMI

Model Klasifikasi

Dalam penelitian ini, dilakukan perbandingan performa empat metode klasifikasi, yaitu *decision tree classifier, random forest classifier, gradient boosting classifier, dan XGBoost classifier.* Perbandingan performa dilakukan sebelum dan setelah dilakukan *tuning hyperparameter*. *Tuning hyperparameter* merupakan suatu proses yang bertujuan untuk mendapatkan parameter paling optimal dari suatu model (Siswantoro *et al.,* 2022). *Tuning hyperparameter* dilakukan dengan metode *grid search CV.* Metode *grid search CV* dianggap sebagai metode yang teliti karena melakukan eksplorasi parameter dengan mengatur jenis nilai prediksi terlebih dahulu (Yulianti *et al.,* 2022). Konfigurasi *hyperparameter* optimal dipilih berdasarkan nilai akurasi *cross-validation* tertinggi dari pilihan *hyperparameter* yang diajukan.

1. *Decision Tree Classifier*

Berdasarkan Tabel 3, terlihat hasil akurasi dari klasifikasi menggunakan algoritma *decision tree* sebelum dan setelah *tuning hyperparameter*. Metode *decision tree classifier* menunjukkan nilai akurasi pada *test data* sebesar 0,796 dan pada *train data* sebesar 1 sebelum dilakukan *tuning hyperparameter.* Setelah *tuning hyperparameter*, diperkirakan terdapat empat parameter yang dapat meningkatkan kinerja model dalam melakukan klasifikasi dengan metode *decision tree*. Proses *tuning hyperparameter* dilakukan menggunakan metode *grid search CV*. Hasil nilai *hyperparameter* terbaik ditampilkan pada Tabel 4.

Tabel 3. Nilai akurasi algoritma *decision tree classifier*

Metode klasifikasi	Akurasi sebelum *tuning hyperparameter*		Akurasi setelah *tuning hyperparameter*	
	Test data	*Train data*	*Test data*	*Train data*
Decision tree classifier	0,796	1,0	0,837	0,847

Tabel 4. Hasil nilai terbaik untuk *tuning hyperparameter* algoritma *decision tree*

Hyperparameter	*Grid search value*	Nilai *hyperparameter* Terbaik
criterion	entropy, gini, log_loss	entropy
splitter	best, random	best
max_depth	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	5
min_sample_leaf	6	6

Setelah memperoleh nilai parameter terbaik yang tercatat dalam Tabel 2, konfigurasi parameter tersebut kemudian digunakan untuk melatih ulang model *decision tree.* Dari Tabel 1, terlihat bahwa setelah dilakukan *tuning hyperparameter*, model dengan metode *decision tree classifier* menunjukkan peningkatan nilai akurasi pada *test data* menjadi 0,837, sedangkan pada *train data* terjadi penurunan nilai akurasi menjadi 0,847. Selisih akurasi antara *train data* dan *test data* semakin kecil setelah *tuning hyperparameter*, menunjukkan bahwa *tuning hyperparameter* memperbaiki model sehingga model tidak terjadi *overfitting*.

2. Random Forest Classifier

Berdasarkan Tabel 5, terlihat hasil akurasi klasifikasi menggunakan algoritma *random forest* sebelum dan setelah *tuning hyperparameter*. Pada metode *random forest classifier,* ditemukan bahwa nilai akurasi pada *test data* sebelum *tuning hyperparameter* adalah 0,898 sementara pada *train data* mencapai 1. Setelah dilakukan *tuning hyperparameter,* diperkirakan terdapat tiga parameter yang dapat meningkatkan kinerja model dalam melakukan klasifikasi dengan metode *random forest*. Proses *tuning hyperparameter* dilakukan menggunakan metode *grid search CV*. Hasil nilai *hyperparameter* terbaik ditampilkan pada Tabel 6.

Tabel 5. Nilai akurasi algoritma *random forest classifier*

Metode klasifikasi	Akurasi sebelum *tuning hyperparameter*		Akurasi setelah *tuning hyperparameter*	
	Test data	*Train data*	*Test data*	*Train data*
Random forest classifier	0,898	1,0	0,877	0,942

Tabel 6. Hasil nilai terbaik untuk *tuning hyperparameter* algoritma *random forest*

Hyperparameter	*Grid search value*	Nilai *hyperparameter* Terbaik
n_estimators	1, 2, 3, ..., 100	18
criterion	entropy, gini, log_loss	entropy
max_features	sqrt, log2, None	None

Setelah mendapatkan nilai parameter terbaik yang dicatat dalam Tabel 4, konfigurasi parameter tersebut kemudian digunakan untuk melatih ulang model *random forest.* Dari Tabel 3, terlihat bahwa setelah dilakukan *tuning hyperparameter,* model dengan metode *random forest classifier* menunjukkan penurunan nilai akurasi pada *test data* menjadi 0,877 dan *train data* menjadi 0,942. Perbedaan nilai akurasi pada *test data* dan *train data* yang masih cukup besar mengindikasikan terjadinya *overfitting* pada model. Model *overfitting* cenderung mengingat semua data, termasuk *noise* pada *training-set*, bukannya memahami pola yang mendasari data. Untuk mengatasi ini, algoritma efektif harus dapat membedakan data representatif dari *noise* (Ying, 2018). Faktor-faktor penyebab *overfitting* melibatkan kompleksitas hipotesis yang mempengaruhi keseimbangan antara akurasi dan konsistensi model. Selain itu, prosedur perbandingan ganda yang umum dalam algoritma induksi dan kecerdasan buatan (AI) dapat memilih item dengan skor maksimum, tetapi juga berpotensi memilih item yang tidak meningkatkan akurasi klasifikasi (Ying, 2018). Oleh karena itu, penting untuk mencapai keseimbangan yang tepat dalam pengembangan model untuk mencegah *overfitting.*

3. Gradient Boosting Classifier

Berdasarkan Tabel 7, terlihat hasil akurasi klasifikasi menggunakan algoritma *gradient boosting* sebelum dan setelah *tuning hyperparameter*. Pada metode *gradient boosting classifier,* ditemukan bahwa nilai akurasi pada *test data* sebelum *tuning hyperparameter* adalah 0,816, sementara pada *train data* mencapai 1. Setelah dilakukan *tuning hyperparameter*, diperkirakan terdapat tiga parameter yang dapat meningkatkan kinerja model dalam melakukan klasifikasi dengan metode *gradient boosting.* Proses *tuning hyperparameter* dilakukan menggunakan metode *grid search CV.* Hasil nilai *hyperparameter* terbaik ditampilkan pada Tabel 8.

Tabel 7. Nilai akurasi algoritma *gradient boosting classifier*

Metode klasifikasi	Akurasi sebelum *tuning hyperparameter*		Akurasi setelah *tuning hyperparameter*	
	Test data	*Train data*	*Test data*	*Train data*
Gradient boosting classifier	0,816	1,0	0,867	0,966

Tabel 8. Hasil nilai terbaik untuk *tuning hyperparameter* algoritma *gradient boosting*

Hyperparameter	*Grid search value*	Nilai *hyperparameter* terbaik
loss	exponential, log_loss	log_loss

criterion	friedman_mse, squared_error	friedman_mse
learning_rate	0,01; 0,025	0,025

Setelah mendapatkan nilai parameter terbaik yang dicatat dalam Tabel 6, konfigurasi parameter tersebut kemudian digunakan untuk melatih ulang model *gradient boosting*. Dari Tabel 5, terlihat bahwa setelah dilakukan *tuning hyperparameter*, model dengan metode *gradient boosting classifier* menunjukkan nilai akurasi meningkat pada *test data* menjadi 0,867 dan menurun pada *train data* menjadi 0,966. Hal ini menandakan bahwa model ini mengalami perbaikan kinerja setelah dilakukan *tuning hyperparameter*.

4. XGBoost Classifier

Dilihat dari Tabel 9, tergambar hasil akurasi klasifikasi dengan menerapkan algoritma *XGBoost* sebelum dan sesudah *tuning hyperparameter.* Pada *XGBoost classifier*, terdapat pencapaian nilai akurasi pada *test data* sebesar 0.837 sebelum *tuning hyperparameter,* sementara pada *train data* mencapai 1. Perbedaan nilai akurasi *test data* dan *train data* yang jauh mengindikasikan terjadinya *overfitting* pada model sehingga perlu dilakukan *tuning hyperparameter.* Setelah proses *tuning hyperparameter*, teridentifikasi adanya tujuh parameter yang berpotensi meningkatkan kinerja model dalam melakukan klasifikasi dengan metode *XGBoost*. Proses *tuning hyperparameter* dilakukan dengan merujuk pada nilai parameter dari jurnal dan menggunakan metode *grid search CV* untuk menentukan nilai *n_estimator.* Hasil nilai *hyperparameter* terbaik ditampilkan pada Tabel 10.

Tabel 9. Nilai akurasi algortima *XGBoost classifier*

Metode klasifikasi	Akurasi sebelum *tuning hyperparameter*		Akurasi setelah *tuning hyperparameter*	
	Test data	*Train data*	*Test data*	*Train data*
XGBoost classifier	0,837	1,0	0,837	0,857

Tabel 10. Hasil nilai terbaik untuk *tuning hyperparameter* algoritma *XGBoost*

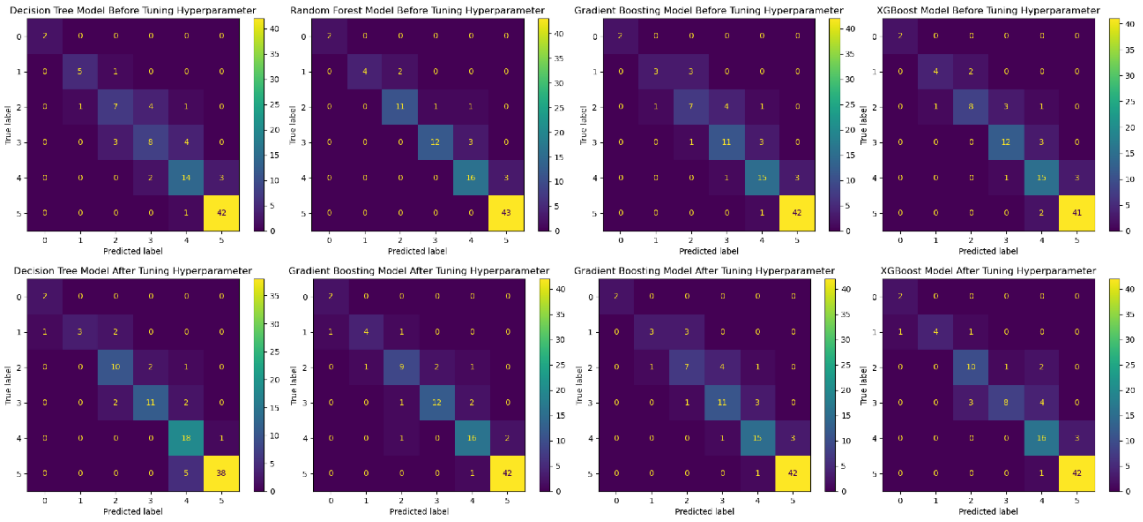
Hyperparameter	Nilai	Sumber
max_depth	6	Yulianti *et al.,* (2022)
learning_rate	0,025	Yulianti *et al.,* (2022)
n_estimators	150	*Grid Search CV*
min_child_weight	7	Yulianti *et al.,* (2022)
colsample_bylevel	1	Yulianti *et al.,* (2022)
subsample	1	Yulianti *et al.,* (2022)
gamma	2	Yulianti *et al.,* (2022)

Setelah memperoleh nilai parameter terbaik yang tercatat dalam Tabel 8, konfigurasi parameter tersebut kemudian diterapkan untuk melatih ulang model *XGBoost.* Dari Tabel 7, dapat diamati bahwa setelah dilakukan *tuning hyperparameter*, model dengan metode *XGBoost classifier* menunjukkan nilai akurasi *test data* tetap pada 0,837. Sedangkan, akurasi pada *train data* mengalami perbedaan menjadi 0,857. Perbedaan pada akurasi *train data* menunjukkan bahwa *tuning hyperparameter* memberikan dampak positif pada model *XGBoost Classifier* karena dapat mengatasi *overfitting* yang terjadi saat sebelum *tuning hyperparameter*. Hal ini ditunjukkan dengan selisih nilai akurasi antara *train data* dan *test data* yang kecil.

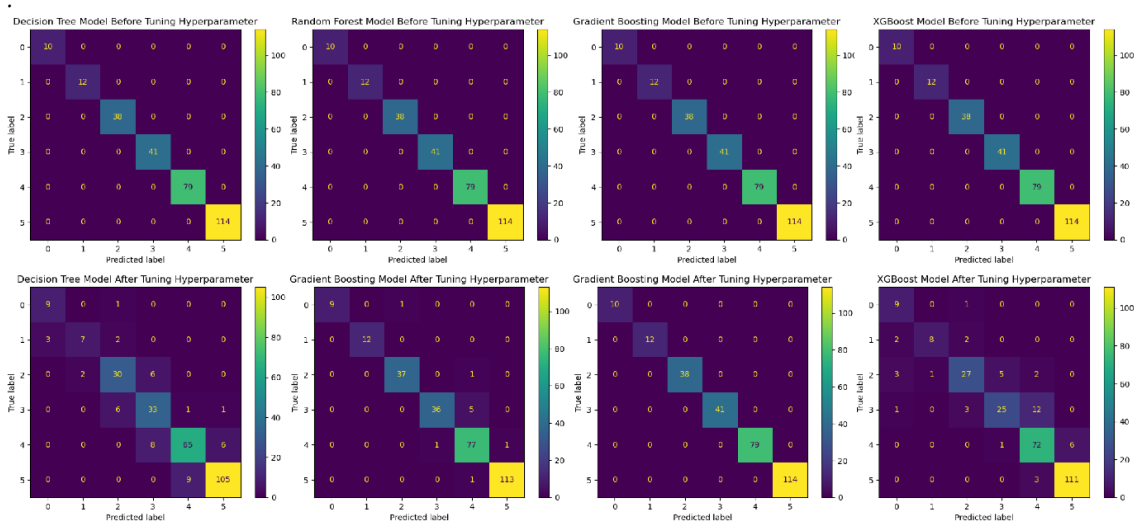
Evaluasi Model

Evaluasi model merupakan langkah krusial dalam *machine learning*. Salah satu metode evaluasi yang umum digunakan adalah *confusion matrix*. *Confusion matriks* merupakan tabel yang menyajikan gambaran menyeluruh tentang kinerja model dengan memperhitungkan empat metrik utama, yaitu *true positive (TP), true negative (TN), false positive (FP), dan false negative (FN)* (Krstinić *et al.,* 2020). Setiap sel di dalam *confusion matrix* mewakili jumlah prediksi yang dilakukan oleh model dibandingkan dengan kelas sebenarnya dari data tersebut. Pada penelitian kali ini, dibuat *confusion matrix* untuk empat model *machine learning,* yaitu *decision tree, random forest, gradient boosting,* dan *XGBoost*. *Confusion matrix* setiap model dibuat dalam dua kondisi, yaitu sebelum *tuning hyperparameter* dan setelah *tuning hyperparameter.* Gambar 6 dan Gambar 7 menunjukkan *confusion matrix* untuk *train*

data dan *test data* saat sebelum dan sesudah *tuning hyperparameter.*



Gambar 6. Perbandingan *confusion matrix* untuk *train data* saat sebelum dan sesudah *tuning hyperparameter*



Gambar 7. Perbandingan *confusion matrix* untuk *test data* saat sebelum dan sesudah *tuning hyperparameter*

Selanjutnya performa model dihitung menggunakan persamaan 23-29 untuk *test data* dan *train data* dari *confusion matrix* yang dihasilkan pada Gambar 6 dan Gambar 7. Performa model diukur dari *macro average* dari *true positive rate (TPR), positive predicted value (PPV),* dan *F1-score*. Digunakan pendekatan makro sehingga dapat menilai performa model secara lebih umum (Nurdiati *et al.,* 2022). *True Positive Rate (TPR)* atau *recall* menunjukkan seberapa sering model memprediksi positif ketika kelas aktualnya positif. *Positive predicted value (PPV)* atau *precision* menunjukkan seberapa sering model memprediksi benar, ketika model memprediksi positif. Sementara *F1-score* merupakan rataan harmonik dari *recall (TPR)* dan *precision (PPV)* (Nurdiati, 2022).

Tabel 11. Performa model dengan pendekatan makro sebelum *tuning hyperparameter*

Jenis data	*Decision Tree*			*Random Forest*			*Gradient Boosting*			*XGBoost*		
	TPR	*PPV*	*F1*	*TPR*	*PPV*	*F1*	*TPR*	*PPV*	*F1*	*TPR*	*PPV*	*F1*
Train	100%	100%	100%	100%	100%	100%	83%	80%	100%	100%	100%	100%
Test	77%	78%	77%	86%	92%	88%	76%	79%	77%	80%	83%	81%

Tabel 12. Performa model dengan pendekatan makro setelah *tuning hyperparameter*

Jenis data	*Decision Tree*			*Random Forest*			*Gradient Boosting*			*XGBoost*		
	TPR	*PPV*	*F1*	*TPR*	*PPV*	*F1*	*TPR*	*PPV*	*F1*	*TPR*	*PPV*	*F1*
Train	80%	80%	80%	90%	90%	90%	95%	98%	96%	86%	86%	85%
Test	81%	82%	79%	83%	82%	82%	76%	79%	81%	80%	82%	79%

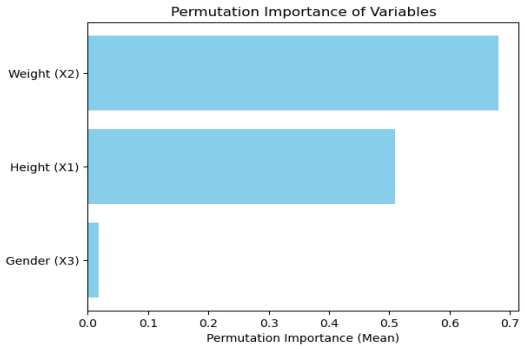
Berdasarkan Tabel 11 terlihat bahwa model *Random Forest* memiliki performa yang lebih baik dibanding model lainnya. Hal ini terlihat dari nilai *F1-score test data* untuk model *Decision Tree, Random Forest, Gradient Boosting, dan XGBoost* masing-masing adalah 77%, 88%, 77%, dan 81%. Namun, nilai *F1-score train data* untuk setiap model masih jauh berbeda dari *F1-score test data*. Hal ini mengindikasikan terjadinya *overfitting* pada setiap model sehingga perlu dilakukan *tuning hyperparameter* untuk menghasilkan model yang optimal. Setelah dilakukan *tuning hyperparameter* model *XGBoost* dan *Decision Tree* menghasilkan model yang tidak *overfit* seperti yang terlihat pada Tabel 12. Namun, meskipun model *Decision Tree* tidak *overfit*, tetapi model *Decision Tree* memiliki nilai *train data* yang rendah diantara model lain. Oleh karena itu, dipilih model *XGBoost* dengan *tuning hyperparameter* sebagai model yang lebih baik karena memiliki sifat yang *robust* terhadap *overfitting* dan performanya juga cukup baik (Yulianti *et al.*, 2022).

Signifikansi Variabel Prediktor

Selanjutnya dilakukan analisis *permutation importance* pada model terbaik yaitu model yang menggunakan algoritma *XGBoost Classifier* sehingga didapatkan hasil seperti pada Tabel 13 dan Gambar 8. *Permutation importance* merupakan suatu metode yang digunakan untuk memutuskan signifikansi variabel prediktor terhadap variabel respon sehingga dapat meningkatkan interpretabilitas model (Altmann *et al.,* 2010). Berdasarkan Tabel 11, variabel berat badan menunjukkan signifikansi tertinggi, berat badan memiliki pengaruh sekitar 68,1% terhadap BMI. Sebaliknya, variabel gender memiliki signifikansi terendah, gender memiliki pengaruh 1,8% terhadap BMI. Di sisi lain, variabel tinggi badan memiliki pengaruh 50.9% terhadap BMI. Oleh karena itu, berdasarkan analisis *permutation importance*, berat badan memiliki pengaruh paling signifikan terhadap indeks BMI, diikuti oleh tinggi badan, sementara gender memiliki pengaruh yang lebih rendah.

Tabel 13. Analisis *permutation importance* algoritma *XGBoost*

Signifikansi *(Importance)*	Tinggi Badan	Berat Badan	Gender
Rata-rata (*mean*)	0,5092	0,6806	0,0184
Standar deviasi	0,0246	0,0158	0,0439



Gambar 8. Plot *permutation importance of variables*

SIMPULAN

Model *Extreme Gradient Boosting (XGBoost)* setelah *tuning hyperparameter* merupakan model terbaik dalam memprediksi kelas BMI individu karena memiliki akurasi yang cukup besar dan tidak *overfit*. Dengan nilai akurasi *testing data* sebesar 0.837 dan *train data* sebesar 0,857. Selain itu, model *XGBoost* setelah *tuning* juga menghasilkan performa terbaik dengan nilai *F1-score test data* 85% dan 79%. Dengan menggunakan model *XGBoost* yang memiliki performa terbaik, diketahui bahwa berat badan merupakan yang paling berpengaruh dalam mengklasifikasikan indeks BMI dengan nilai signifikansi sebesar 0,6806 dibandingkan dengan tinggi badan dan jenis kelamin yang secara berturut-turut memiliki nilai signifikansi sebesar 0,5076 dan 0,0906.

DAFTAR PUSTAKA

Abdurrahman, G., Oktavianto, H., & Sintawati, M. (2022). Optimasi algoritma XGBoost classifier menggunakan hyperparameter grid search dan random search pada klasifikasi penyakit diabetes. *Informatics Journal*, 7(3), 193-198. https://doi.org/10.19184/isj.v7i3.35441.

Altmann, A., Tolosi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10), 1340-1347. https://doi.org/10.1093/bioinformatics/btq134.

Heath, M.T. (2002). *Scientific Computing: An Introduction to Survey*. Ed ke-2. New York: McGraw-Hill.

Kulkarni, V. Y., & Sinha, P. K. (2013). Random forest classifier: a survey and future research directions. *International Journal of Advanced Computing*, 36(1), 1144-1153.

Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Ed ke-1. New Jersey: John Wiley & Sons, Inc..

Krstinić, D., Braović, M., Šerić, L., & Štulić, D. B. (2020). Multi-label classifier performance evaluation with confusion matrix. *Computer Science & Information Technology (CS & IT)*. https://doi.org/10.5121/csit.2020.100801.

Kunanbayev, K., Temirbek, I., & Zollanvari, A. (2021). Complex encoding. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN52387.2021.9534094.

Lavanya, D., & Rani, K. U. (2012). Ensemble decision tree classifier for breast cancer data. *International Journal of Information Technology Convergence and Services (IJITCS),* 2(1), 17-24. https://doi.org/10.5121/ijitcs.2012.2103.

Leni, D., Arwizet, K., Sumiati, R., Haris, H., & Adriansyah, A. (2023). Perancangan metode *machine learning* berbasis web untuk prediksi sifat mekanik aluminium. *Jurnal Rekayasa Mesin*, 14(2), 611-626.

Murbawani, E. A., Puruhita, N., Yudomurti, Y. (2012). Tinggi badan yang diukur dan berdasarkan tinggi lutut menggunakan rumus *chumlea* pada lansia. *Media Medika Indonesian*, 46(1), 1–6.

Nagasaka, K., Harada, N., Ichihashi, H., & Leonard, R. (1994). Adaptive learning networks of multi-stage fuzzy production rules in expert system of grinding characteristics. *Comput Ind Eng*, 27(1–4), 433–436. https://doi.org/10.1016/0360-8352(94)90327-1.

Nurdiati, A., Najib, M. K., Bukhari, F., Ardhana, M. R., Rahmah, S., & Blante, T. P. (2022). Perbandingan Alex Net dan VGG untuk pengenalan ekspresi wajah pada dataset kelas komputasi lanjut. *Techno.COM*, 21(3), 500-510.

Peng, Y., & Nagata, M. H. (2020). An empirical overview of nonlinearity and overfitting in machine learning using COVID-19 data. *Chaos, Solitons and Fractals*, 139, 1-16. https://doi.org/10.1016/j.chaos.2020.110055.

Rahman, F., Fauzi, H., Azhar, T. N., Atmadja, R. D., & Ayudina, N. (2017). Analisa metode pengukuran berat badan manusia dengan pengolahan citra. *Teknik*, 38(1), 35–39.

Ramadani, D. (2023). Metode random forest dan xgboost: studi kasus prediksi arah penutupan harga saham [Skripsi]. Bogor (ID): IPB University

Raschka S, Liu YH, Mirjalili V, Dzhulgakov D. 2022. *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Ed ke-1. Birmingham: Packt Publishing Ltd.

[Riskesdas] Kementerian Kesehatan RI. 2018. Hasil Utama Riskesdas. [diakses 2023 Desember 05]. https://kesmas.kemkes.go.id/assets/upload/dir_519d41d8cd98f00/files/Hasil-riskesdas-2018_1274.pdf

Sartono, B., & Syafitri, U. D. (2010). Metode pohon gabungan: solusi pilihan untuk mengatasi kelemahan pohon regresi dan klasifikasi tunggal. *Forum Statistika Komputasi*, 15(1), 1-7.

Siswantoro, M. Z. F. N., & Yuhana, U. L. (2023). Software defect prediction based on optimized machine learning models: a comparative study. *TEKNIKA*, 12(2), 166-172. https://doi.org/10.34148/teknika.v12i2.634.

Sriningsih, M., Hatidja, D., & Prang, J. D. (2018). Penanganan multikolinearitas menggunakan analisis regresi komponen utama pada kasus impor beras di Provinsi Sulut. *Jurnal Ilmiah Sains*, 18(1), 19-23.

[SSGI] Kementerian Kesehatan RI. 2022. Buku Saku Hasil Survei Status Gizi Indonesia. [diakses 2023 Desember 05]. https://kesmas.kemkes.go.id/assets/uploads/contents/attachments/09fb5b8ccfdf088080f2521ff0b4374f.pdf.

Suryana, S. E., Warsito, B., & Suparti, S. (2021). Penerapan gradient boosting dengan hyperopt untuk memprediksi keberhasilan telemarketing bank. *JURNAL GAUSSIAN*, 10(4), 617-623.

[UNICEF] Forum Anak Nasional. 2022. Analisis Lanskap Kelebihan Berat Badan dan Obesitas di Indonesia. [diakses 2023 Desember 05]. https://www.unicef.org/indonesia/media/15581/file/Analisis%20Lanskap%20Kelebihan%20Berat%20Badan%20dan%20O besitas%20di%20Indonesia.pdf.

Wainer, J., & Cawley, G. (2017). Empirical evaluation of resampling procedures for optimising SVM hyperparameters. *J Mach Learn Res*, 18(13), 1–35.

Wibowo, R. A., & Kurniawan, A. A. (2020). Analisis korelasi dalam penentuan arah faktor pada pelayanan angkutan umum di Kota Magelang. *Journal of Electrical Engineering, Computer and Information Technology*. https://doi.org/10.31002/jeecit.v1i2.3552.

Wijayanti, N. P. Y. T., Kencana, E. N., & Sumarjaya, I. W. (2021). SMOTE: potensi dan kekurangannya pada survei. *E-Jurnal Matematika*, 10(4), 235-240. https://doi.org/10.24843/MTK.2021.v10.i04.p348.

Wulandari, M. Z., Hamdi, A. F, Nurhalisa, F. Z., Hutabarat, D. F., Septiani, G. C., Nurazizah, D. A., Puspadewi, S. (2023). Penggunaan perhitungan indeks massa tubuh sebagai penanda status gizi pada mahasiswa prodi kesehatan masyarakat Rombel 2D. *Jurnal Analis*, 2(2), 124–131.

Ying, X. (2018). An overview of overfitting and it's solutions. *Journal of Physics: Conference Series*, 1168(2), 1-6. https://doi.org/10.1088/1742-6596/1168/2/022022.

Yulianti, S. E. H., Soesanto, O., & Sukmawaty, Y. (2022). Penerapan metode extreme gradient boosting (XGBoost) pada klasifikasi nasabah kartu kredit. *Journal of Mathematics: Theory and Applications*, 4(1), 21-26.