Studi Komparasi Performa Algoritma *Supervised-Learning* dalam Klasifikasi Multidataset

I Gusti Ngurah Ryo Aditarta
Fakultas Ilmu Komputer
Universitas Brawijaya
Malang, Indonesia
ryoaditarta@student.ub.ac.id

2nd Dian Pandu Syahfitra
Fakultas Ilmu Komputer
Universitas Brawijaya
Malang, Indonesia
dianpandu@student.ub.ac.id

3rd Rayhan Egar Sadtya Nugraha
Fakultas Ilmu Komputer
Universitas Brawijaya
Malang, Indonesia
rayhanegar@student.ub.ac.id

4th Ahmad Zaki
Fakultas Ilmu Komputer
Universitas Brawijaya
Malang, Indonesia
ahmadzaki12@student.ub.ac.id

5th Muhammad Arya Ghifari
Fakultas Ilmu Komputer
Universitas Brawijaya
Malang, Indonesia
aryaghifary07@student.ub.ac.id

6th Arion Syemael Siahaan
Fakultas Ilmu Komputer
Universitas Brawijaya
Malang, Indonesia
siahaanarion@student.ub.ac.id

***Abstract*—Machine Learning (ML) is a cornerstone technology advancing various aspects of human life, focusing on developing algorithms that enable systems to learn from data and make predictions or decisions without explicit programming. A key aspect of ML is supervised learning, where models learn from input-output pairs to predict or classify new data. This paper aims to address the challenge of selecting appropriate classification methods by comparing various techniques from the literature.**

Datasets used include Spaceship Titanic, Bank Churn, Horse Health, and Keep-it-Dry, each chosen for their diverse features and predictive challenges. Preprocessing steps involve data encoding, scaling, and handling missing values. Supervised learning algorithms explored are CatBoost, AdaBoost, LightGBM, and ensemble methods like stacking and K-Nearest Neighbor (KNN). Hyperparameter optimization techniques such as GridSearchCV and Optuna are employed to enhance performance. Evaluation metrics include confusion matrix, recall, precision, accuracy, F1-score, AUC-ROC, and cross-validation.

Results indicate that tree-based classifiers like LightGBM, CatBoost, and XGBoost generally perform well across different datasets, achieving higher accuracy and better generalization. However, linear classifiers like Ridge Classifier and Logistic Regression also show competitive performance, especially in stability and reliability on test data, when combined with appropriate preprocessing techniques.

In conclusion, this study underscores the importance of proper data preprocessing, the effectiveness of Principal Component Analysis (PCA) in improving model generalization, and the critical role of hyperparameter tuning in developing robust ML models. The insights from this comparative analysis can guide practitioners in selecting and fine-tuning classification algorithms for varied predictive tasks.

***Index Terms*—Supervised Machine Learning, Classification Algorithms, Supervised Machine Learning Comparative Study**

I. Introduction

Pembelajaran Mesin (ML) telah menjadi salah satu fondasi teknologi yang mendukung kemajuan teknologi dalam berbagai aspek kehidupan manusia [2]. Sebagai cabang dari kecerdasan buatan (AI), ML bertujuan untuk mengembangkan algoritma dan model komputasional yang memungkinkan sistem untuk belajar dari data dan membuat prediksi atau keputusan tanpa perlu diprogram secara eksplisit [1].

Salah satu aspek penting dari ML adalah supervised control, di mana model diberikan pasangan input-output dalam data pelatihan untuk mempelajari hubungan antara mereka [1]. Dengan menggunakan metode ini, model dapat melakukan prediksi atau klasifikasi pada data baru berdasarkan pembelajaran dari data pelatihan yang ada [1].

Klasifikasi dengan supervised learning adalah salah satu teknik yang paling fundamental dan sering digunakan [1]. Dengan memanfaatkan dataset yang telah dilabeli, model dilatih untuk mengenali pola dan membuat prediksi yang akurat terhadap data baru [2]. Namun, dalam menghadapi berbagai tugas klasifikasi, peneliti dan praktisi ML seringkali dihadapkan pada tantangan dalam memilih metode yang paling sesuai untuk kasus tertentu [1].

Paper ini dibuat bertujuan untuk menjawab tantangan tersebut dengan melakukan studi perbandingan terhadap berbagai metode klasifikasi yang telah diusulkan dalam literatur ilmiah [3]. Dengan menggunakan referensi dari jurnal-jurnal terkemuka, kami mengumpulkan berbagai pendekatan klasifikasi dengan supervised yang memiliki keunggulan dan kelemahan masing-masing [3].

II. Algoritma Klasifikasi Supervised Learning

A. K-Nearest Neighbors (KNN)

Algoritma klasifikasi *nearest neighbors* seperti KNN merupakan salah satu algoritma dasar nonparametrik dalam pembelajaran mesin. Hal ini didasarkan pada *rationale* di mana fitur yang digunakan untuk mendeskripsikan label sebuah *domain point* memiliki relevansi dengan *domain point* lain dalam sisi *proximity* [4], [5]. Dalam implementasinya, KNN banyak digunakan dalam permasalahan klasifikasi dengan domain pengetahuan yang terdefinisi dan diketahui dengan baik, seperti klasifikasi tumor otak [6], klasifikasi tingkat keparahan Covid-19 [7], maupun prognosis kanker [8].

Secara prinsip, nilai parameter k yang merepresentasikan banyaknya *domain point* yang digunakan dalam penentuan label serta metode perhitungan jarak merupakan konsiderasi utama. Nilai k yang terlalu kecil memberikan model yang kompleks dan kurang mampu mengakomodasi *unseen data*, sedangkan nilai k yang terlalu besar memberikan model yang terlalu sederhana untuk secara akurat mengklasifikasikan suatu *domain point* [9]. Beberapa teknik, seperti *normalized class coherence*, *change-based KNN*, *variable selection* dan *weighting*, maupun normalisasi L1 dan LPP dapat digunakan untuk memberikan estimasi yang lebih baik [10]–[12]. Selain itu, pilihan perhitungan jarak antara domain point dalam proses pembelajaran model seperti Euclidean "(1)", Manhattan "(2)", maupun Minkowski "(3)" untuk pemetaan label kelas juga perlu untuk diperhatikan.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \tag{1}$$

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i| \tag{2}$$

$$d(x,y) = (\sum_{i=1}^{n}|x_i - y_i|^p)^{\frac{1}{4}} \tag{3}$$

B. Ridge Classifier (RC)

Algoritma klasifikasi *Ridge Classifier* (RC) dikembangkan dari algoritma *Ridge Regression* yang mengkombinasikan *learning rule Regularized Learning Minimization* (RLM) dengan regresi *linear ordinary least squares* [1]. Algoritma RC melakukan *class labelling* berdasarkan tanda/*sign* dari suatu *data point* (positif atau negatif). Penggunaan fungsi regularisasi, seperti regularisasi L2/Tikhonov "(4)" "(5)" pada algoritma klasifikasi memberikan model yang lebih "stabil" dan mencegah terjadinya *overfitting* [4], [5]. Dibandingkan dengan regresi linear, algoritma RC akan meminimalisasi nilai koefisien w untuk masing-masing fitur, memberikan model dengan kemampuan generalisasi yang baik terhadap *unseen data*. Dengan demikian, fungsi yang digunakan dalam RC dapat didefinisikan secara formal pada "(6)".

$$||w|| = \sqrt{\sum_{i=1}^{d} w_i^2} \tag{4}$$

$$\arg min_w(L_s(w) + \lambda||w||^2) \tag{5}$$

$$\arg min_{w \in R^d}(\lambda||w||^2 + \frac{1}{m}\sum_{i=1}^{m}(< w, x_i > -y_i)^2) \tag{6}$$

Dalam melakukan proses pembelajaran model, parameter regularisasi menjadi konsiderasi utama. Semakin kecil nilai, nilai koefisien w akan semakin kecil, memberikan model dengan generalisasi yang baik namun dengan potensi *underfitting*. Nilai α yang semakin besar akan meminimalisasi efek regularisasi, memberikan model dengan kompleksitas yang lebih tinggi namun dengan potensi *overfitting* untuk *unseen data*. Selain dengan menggunakan *cross-validation* untuk *hyperparameter tuning*, teknik seperti *fractional ridge regression* [13] dengan memanfaatkan rasio L2-norm antara *regularized* dan *normal coefficients* dapat membantu menemukan nilai α yang optimal.

C. Logistic Regression (LR)

Logistic Regression (LR) merupakan algoritma klasifikasi yang didefinisikan secara formal sebagai komposisi fungsi sigmoid "(7)"terhadap suatu fungsi regresi linear untuk membuat kelas hipotesis "(8)" [4]. Algoritma LR, bersama dengan RC, digunakan dalam kasus klasifikasi dengan memperhatikan tanda/*sign* suatu *domain point* [9]. LR memanfaatkan regularisasi L2/*ridge* dengan koefisien regularisasi α, di mana semakin rendah nilai α, model yang dihasilkan akan lebih sederhana dengan koefisien w yang semakin kecil. Sebagai *weak learner*, *ensembling* dari model LR dengan menggunakan AdaBoost dapat membantu meningkatkan performa generalisasi model dengan proses iteratif. Studi [14] menunjukkan jika penggunaan metode *robust functional principal component analysis* (RFPCA) memberikan dampak positif pada performa klasifikasi model LR. Dengan demikian, *learning rule* algoritma klasifikasi LR dapat secara formal didefinisikan dalam bentuk "(9)"

$$\sigma_{sig}(z) = \frac{1}{1 + \exp(-z)} \tag{7}$$

$$H_{sig} = \sigma_{sig} \circ L_d \tag{8}$$

$$\arg min_{w \in R^d}\frac{1}{m}\sum_{i=1}^{m}\log(1 + \exp(-y_i < w, x_i >)) \tag{9}$$

D. Decision Tree (DT)

Decision Tree (DT) merupakan algoritma klasifikasi yang terdiri dari himpunan pertanyaan *if-else* sebagai *splitting criteria* yang dibangun secara iteratif dengan pendekatan *top-down* untuk memisahkan suatu kelas dari kelas lainnya. Setiap node yang terbentuk pada DT memiliki nilai entropi "(10)" yang merepresentasikan rata-rata informasi yang diperlukan untuk melakukan separasi label kelas [15]. Fitur yang dipilih sebagai *splitting criteria* sebuah node untuk menghasilkan *child node* didasarkan atas *gain measure* yang berbeda-beda untuk setiap algoritma DT [4]. Algoritma ID3 mengimplementasikan *Information Gain* (IG) "(11)" sebagai *gain measure*, sedangkan algoritma C4.5 mengimplementasikan Gain Ratio (GR) "(12)" [16]. Pemilihan hyperparameter DT yang optimal untuk suatu dataset, seperti *maximum depth* (MD), *maximum leaf nodes* dan *minimum samples* dalam pembentukan *leaf nodes* mampu memberikan model DT dengan performa generalisasi yang baik dan mencegah *overfitting* akibat *tree size* yang terlalu besar [17].

$$H(S) = \sum_{i=1}^{C} p_i \log_2(p_i) \tag{10}$$

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v) \tag{11}$$

$$GR(S, A) = \frac{IG(S, A)}{SI(S, A)} \tag{12}$$

$$SI(S, A) = - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2(\frac{|S_v|}{|S|}) \tag{13}$$

E. Naive-Bayes (NB)

Algoritma Naive-Bayes adalah salah satu algoritma pembelajaran mesin yang populer untuk tugas klasifikasi. Algoritma ini didasarkan pada teorema Bayes dan beroperasi dengan asumsi "*naive*" bahwa semua fitur yang ada bersifat independen satu sama lain. Asumsi ini menyederhanakan perhitungan probabilitas, menjadikan algoritma ini sangat efisien meskipun dalam kenyataannya fitur-fitur tersebut mungkin tidak sepenuhnya independen [19]. Dalam konteks data fungsional, Naive-Bayes digunakan untuk mengklasifikasikan objek berdasarkan data pelatihan dengan menggunakan "*surrogate densities*" yang diturunkan dari skor *Functional Common Principal Component* (FCPC) [18].

Algoritma Naive-Bayes juga menghadapi tantangan dalam ruang berdimensi tinggi, di mana fungsi kepadatan probabilitas seringkali tidak ada sehingga pendekatan densitas klasik tidak dapat digunakan. Untuk mengatasi masalah ini, asumsi *naive* diterapkan pada skor FCPC yang memungkinkan definisi densitas dari data fungsional [18]. Studi simulasi dan aplikasi pada data nyata menunjukkan bahwa Naive-Bayes sering memberikan performa yang kompetitif dibandingkan dengan algoritma klasifikasi lainnya seperti regresi logistik multinomial, k-NN, analisis diskriminan linear, dan mesin vektor pendukung, terutama ketika jumlah komponen meningkat [18].

- Probabilitas posterior

$$P(C_k|x) = \frac{P(x|C_k) \cdot P(C_k)}{P(x)} \tag{14}$$

- Probabilitas kondisional

$$P(x|C_k) = \prod_{i=1}^{n} P(x_i|C_k) \tag{15}$$

- Estimasi probabilitas dengan distribusi normal (Gaussian Naive-Bayes)

$$C_k = \arg\max_{C \in C} P(C|x) = \arg\max_{C \in C} P(C) \cdot \prod_{i=1}^{n} P(x_i|C) \tag{16}$$

- Klasifikasi dengan Naive-Bayes

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot \exp\left(\frac{-(x_i - \mu_k)^2}{2\sigma_k^2}\right) \tag{17}$$

F. Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi [21]. Algoritma ini bekerja dengan mencari *hyperplane* terbaik yang memisahkan data ke dalam kelas yang berbeda [20]. *Hyperplane* adalah batas keputusan yang memisahkan set data dengan label yang berbeda; dalam ruang dua dimensi, *hyperplane* adalah garis; dalam ruang tiga dimensi, *hyperplane* adalah bidang; dan dalam dimensi yang lebih tinggi, *hyperplane* adalah objek dengan dimensi lebih tinggi yang memisahkan data [22].

Pada dasarnya, SVM bertujuan untuk menemukan *hyperplane* yang memaksimalkan margin, yaitu jarak antara *hyperplane* dan titik data terdekat dari setiap kelas [21]. Untuk data yang tidak dapat dipisahkan secara linear, SVM menggunakan fungsi kernel untuk memetakan data ke dimensi yang lebih tinggi di mana data tersebut dapat dipisahkan secara linear [22]. Beberapa fungsi kernel yang umum digunakan adalah kernel linear, kernel polinomial, dan *Radial Basis Function* (RBF) [20].

Rumus penting dalam SVM melibatkan fungsi objektif yang bertujuan meminimalkan norma vektor bobot, dengan syarat bahwa data dapat dipisahkan dengan margin yang maksimal [21]. Untuk data yang tidak dapat dipisahkan secara sempurna, variabel *slack* digunakan untuk mengatasi kasus-kasus ini [22]. Implementasi SVM melibatkan pemilihan fungsi kernel yang sesuai dan penentuan parameter model yang optimal, serta memecahkan masalah optimasi untuk mendapatkan *hyperplane* yang dapat memprediksi label data baru [20].

- Fungsi Linear SVM

$$f(x) = w^T x + b \tag{18}$$

- Margin optimal

$$Margin = \frac{2}{||w||^2} \tag{19}$$

- Fungsi Objektif untuk SVM (data yang dapat dipisahkan secara linear)

$$\min_{w,b} \frac{1}{2} ||w||^2 \tag{20}$$

- Fungsi Objektif untuk SVM (data yang tidak dapat dipisahkan secara linear)

$$\min_{w,b,\xi} \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} \xi_i \tag{21}$$

- Fungsi Kernel

$$K(x_i, x_j) = x_i^T x_j \tag{22}$$

- Kernel Linear

$$K(x_i, x_j) = (x_i^T x_j + c)^d \tag{23}$$

- Kernel RBF

$$K(x_i, x_j) = \exp(-||x_i - x_j||^2/\gamma) \tag{24}$$

G. Random Forest (RF)

Algoritma *Random Forest* (RF) adalah metode pembelajaran *ensemble* yang beroperasi dengan membangun banyak pohon keputusan selama pelatihan dan menghasilkan mode dari kelas (klasifikasi) atau rata-rata prediksi (regresi) dari masing-masing pohon. Dalam konteks pembelajaran *semi-supervised*, algoritma *Co-Forest* memperluas pendekatan Random Forest dengan menggunakan beberapa *classifier* untuk menangani contoh-contoh yang tidak berlabel [23]. Algoritma ini secara iteratif menyempurnakan setiap *classifier* dengan contoh-contoh baru yang diberi label, yang dipilih berdasarkan kepercayaan *classifier* lain dalam *ensemble* [23]. Hasil eksperimen menunjukkan bahwa *Co-Forest* meningkatkan kinerja, terutama ketika proporsi data berlabel rendah, seperti pada dataset biologis [23]. Dengan menggunakan contoh yang tidak berlabel untuk meningkatkan pembelajaran dari sampel yang diberi label, *Co-Forest* menunjukkan peningkatan rata-rata sebesar 3,6% pada kondisi dengan 60% data tidak berlabel [23].

Pendekatan lain yang diperkenalkan adalah *Confidence weighted Random Forest* (CwRF), yang menambahkan skor kepercayaan untuk setiap node daun dalam pohon keputusan [24]. Skor kepercayaan ini digunakan untuk memberi bobot pada suara dari pohon-pohon tersebut, memberikan pengaruh lebih besar pada pohon yang membuat prediksi dengan lebih percaya diri [24]. Kepercayaan dihitung berdasarkan metrik *impurity* seperti entropi dan indeks Gini [24]. Skor kepercayaan ini kemudian digunakan untuk menimbang probabilitas kelas dari setiap pohon selama fase pengujian [24]. Hasil eksperimen menunjukkan bahwa CwRF secara konsisten mengungguli RF tradisional dan metode canggih lainnya pada berbagai dataset, menunjukkan efektivitasnya dalam meningkatkan proses pengambilan keputusan secara keseluruhan [24]. Algoritma ini terbukti efektif di berbagai aplikasi, memperkuat potensinya untuk diterapkan secara lebih luas [24].

- *GINI Impurity* (GI)

$$GI(S) = 1 - \sum_{i=1}^{C} \frac{|S_i|}{|S|}^2 \tag{25}$$

- *Information Gain* (IG)

$$IG(S, A) = GI(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GI(S_v) \tag{26}$$

- *Out-of-Bag Error*

$$OOB_{Error} = \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq f_i(x_i)) \tag{27}$$

H. Extreme Gradient Boosting (XGBoost)

XGBoost (*eXtreme Gradient Boosting*) adalah algoritma pembelajaran mesin berbasis *boosting gradient* yang sangat skalabel [25]. Algoritma ini sering digunakan dalam berbagai kompetisi pembelajaran mesin karena kecepatan pelatihan dan kinerja generalisasinya yang unggul [26]. XGBoost bekerja dengan menggabungkan beberapa model pembelajaran lemah secara iteratif untuk membentuk model yang lebih kuat [26]. Algoritma ini membangun model *ensemble* dari pohon-pohon keputusan menggunakan fungsi aditif untuk meminimalkan fungsi objektif yang teratur [26]. Fungsi objektif dalam XGBoost menggabungkan fungsi *loss* dan penalti untuk menghindari *overfitting*, di mana fungsi *loss* dan regularisasi ini membantu mengontrol kompleksitas model [26]. Parameter penting dalam XGBoost termasuk laju pembelajaran, gamma, kedalaman maksimum, dan subsampling, yang semuanya digunakan untuk mengoptimalkan kinerja model [26].

Untuk mengoptimalkan fungsi objektif, XGBoost menggunakan pendekatan orde kedua, di mana statistik gradien pertama dan kedua pada fungsi *loss* digunakan [26]. Skor untuk pemilihan split dihitung berdasarkan jumlah gradien pertama dan kedua, sementara bobot daun dioptimalkan untuk meminimalkan *loss* [26]. Dengan teknik ini, XGBoost mampu menangani dataset besar dengan efisiensi tinggi dan memberikan kinerja prediksi yang unggul [3]. Algoritma ini juga menggabungkan berbagai teknik seperti regularisasi, subsampling, dan optimasi berbasis cache untuk meningkatkan kecepatan pelatihan dan mengurangi *overfitting* [26]. Keseluruhan, XGBoost adalah algoritma yang sangat efektif dalam pembelajaran mesin dan telah terbukti unggul dalam berbagai tugas klasifikasi dan regresi [25]. XGBoost sangat diakui karena kemampuannya dalam menangani data dalam skala besar dan kompleks, serta memberikan hasil yang sangat akurat dalam berbagai kompetisi pembelajaran mesin [25] [26].

- *Objective function*

$$Obj(t) = \sum_{i=1}^{n} L(t, i) + f(t) \tag{28}$$

- *Regression loss*

$$L(t, i) = \frac{1}{2} \left(y_i - f_t(x_i)\right)^2 \tag{29}$$

- Penalti Regularisasi L1

$$f(t) = \lambda \sum_{j=1}^{K} |w_j| \tag{30}$$

- Penalti Regularisasi L2

$$f(t) = \lambda \sum_{j=1}^{K} w_j^2 \tag{31}$$

- *Gradient*

$$g_i = \partial_{\hat{y}_i} l(y_i, \hat{y}_i) \tag{32}$$

- *Hessian*

$$h_i = \partial^2_{\hat{y}_i} l(y_i, \hat{y}_i) \tag{33}$$

- Formula pembaruan bobot daun

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{34}$$

I. Categorical Boosting (CatBoost)

Algoritma *machine learning* ini digunakan untuk menangani fitur kategorikal dan juga lebih cepat dibandingkan dengan algoritma penguat lainnya karena mengimplementasikan pohon simetris. CatBoost, yang merupakan implementasi dari Gradient Boosting on Decision Tree (GDBT), memiliki kombinasi gradient boosting dengan pohon keputusan yang memberikan hasil yang baik [28].

CatBoost dirancang untuk menangani data kategorikal secara efisien dengan mengkonversinya menjadi fitur numerik secara otomatis. Proses pelatihannya melibatkan pembuatan sejumlah model pohon keputusan secara berurutan, di mana setiap model baru berusaha untuk mengurangi kesalahan dari model sebelumnya. Teknik boosting ini memungkinkan CatBoost untuk memperbaiki kesalahan prediksi secara iteratif. Selain itu, CatBoost menggunakan pohon simetris yang mempercepat waktu prediksi dan pelatihan. Algoritma ini juga memanfaatkan regularisasi untuk menghindari overfitting dan meningkatkan generalisasi model [29].

J. Adaptive Boosting (AdaBoost)

Algoritma Adaboost adalah salah satu dari *boosting classification algorithm* yang dapat meningkatkan kelompok klasifikasi "lemah" menjadi klasifikasi "kuat" [33]. Dalam *supervised learning*, *boosting* digunakan untuk mengurangi bias dan variasi. Prosesnya dimulai dengan melatih model dasar (biasanya decision tree sederhana) pada dataset asli. Setelah setiap iterasi, data yang diklasifikasikan dengan benar oleh model diberi bobot yang lebih rendah, sementara data yang salah diklasifikasikan diberi bobot yang lebih tinggi. Model selanjutnya dilatih untuk fokus pada data yang sulit ini. Proses ini diulang beberapa kali, dan model akhir adalah kombinasi berbobot dari semua model dasar yang dilatih. Bobot setiap model dalam kombinasi akhir ditentukan berdasarkan akurasinya. Dengan demikian, AdaBoost secara adaptif meningkatkan kinerja model dengan memberikan perhatian lebih pada kesalahan yang dibuat oleh model sebelumnya [34].

K. Light Gradient Boosting Machine (LGBM)

Algoritma LightGBM dikembangkan oleh Microsoft yang memberikan kemampuan yang efisien dari algoritma *Gradient Boosting*. LightGBM memiliki karakteristik yang membedakannya dengan algoritma *tree boosting* lainnya adalah dengan membelah pohon secara memanjang (*leaf-wise tree growth*) dengan yang paling cocok, sedangkan algoritma *tree boosting* lainnya membagi pohon secara mendalam atau sejajar (*level-wise tree growth*) yang ditunjukkan pada "Fig. 1" [30].

LightGBM menggunakan dua teknik utama, yaitu *Exclusive Feature Bundling* (EFB) dan *Gradient-based One-Side Sampling* (GOSS), untuk meningkatkan kecepatan dan mengurangi penggunaan memori. EFB menggabungkan fitur-fitur yang jarang digunakan menjadi satu fitur, mengurangi jumlah fitur yang perlu diproses. GOSS, di sisi lain, memilih sampel dengan gradien besar dan mempertahankan distribusi data,

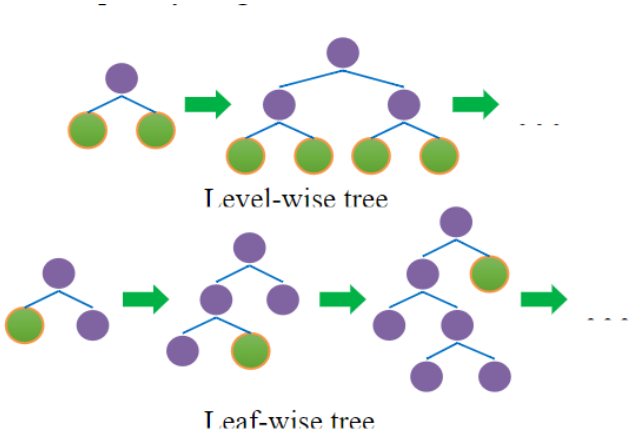

Fig. 1. Light Gradient Boosting Machine (LGBM)

sehingga mengurangi jumlah data yang diproses tanpa kehilangan akurasi. Proses pelatihan LightGBM dimulai dengan membangun pohon keputusan dari data pelatihan, dan setiap iterasi bertujuan untuk mengurangi kesalahan dari model sebelumnya dengan menggunakan gradien untuk mengarahkan perbaikan [31].

L. Ensemble (Stacking)

Stacking termasuk dalam kategori *ensemble learning* yang didalamnya terdapat proses tumpukan dari beberapa algoritma klasifikasi tunggal. Dalam penggunaannya, terdapat dua level model pembelajaran yang akan digunakan ketika metode *stacking* diterapkan, yang pertama adalah model pembelajaran level-0 yang di kenal sebagai *base learner*, dan model pembelajaran level-1 yang merupakan *meta-learner*. Model *Stacking* ini menggunakan model XGBoost, LightGBM, dan CatBoost sebagai *base learner*, dan LR sebagai *meta learner* [32]. Ilustrasi dari proses *stacking* digambarkan melalui "Fig. 2".

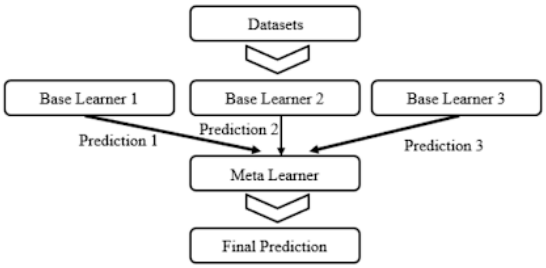

Fig. 2. Ensemble dengan Stacking

III. Dataset

A. Spaceship Dataset

Dataset Spaceship Titanic adalah kumpulan data yang memuat informasi tentang penumpang kapal luar angkasa dan keterkaitannya dengan keputusan mereka untuk diangkut ke

dimensi lain atau tidak. Tujuan dari dataset ini adalah untuk memprediksi apakah seorang penumpang akan diangkut ke dimensi lain atau tidak.

Dataset ini menyediakan fitur-fitur untuk melakukan prediksi. *HomePlanet* adalah planet asal penumpang, *CryoSleep* menunjukkan apakah penumpang akan dibekukan atau tidak, VIP menunjukkan apakah penumpang memiliki layanan VIP atau tidak, *RoomService, FoodCourt, shopping mall, Spa, VRDeck* adalah fitur-fitur yang menunjukkan pengeluaran penumpang, *Transported* adalah target class yang menunjukkan apakah penumpang diangkut ke dimensi lain atau tidak.

Dalam dataset ini, fitur-fitur yang digunakan untuk prediksi meliputi *HomePlanet*, *CryoSleep*, VIP, *RoomService, FoodCourt, shopping mall, Spa, VRDeck*, dan *Transported*. Fitur-fitur ini digunakan untuk memprediksi apakah seorang penumpang akan diangkut ke dimensi lain atau tidak.

Dalam penelitian ini, dataset Spaceship Titanic digunakan untuk memprediksi apakah seorang penumpang akan diangkut ke dimensi lain atau tidak. Dataset ini digunakan untuk mempelajari pola dan hubungan antara fitur-fitur yang digunakan untuk prediksi. Dengan menggunakan dataset ini, penelitian ini dapat membantu dalam meningkatkan akurasi prediksi dan meningkatkan kemampuan sistem dalam memprediksi keputusan penumpang.

B. Bank Churn Dataset

Dataset Bank Churn adalah kumpulan data yang memuat informasi tentang nasabah bank dan keterkaitannya dengan keputusan mereka untuk berhenti menggunakan layanan atau produk yang ditawarkan oleh bank tersebut. Tujuan dari dataset ini adalah untuk memprediksi apakah seorang nasabah akan tetap menggunakan akun mereka atau akan menutupnya.

Dataset ini menyediakan fitur-fitur untuk melakukan prediksi. *Id* dan *customerId* adalah dua fitur yang digunakan sebagai identitas dari *customer* yang sifatnya unik. *Surname* adalah nama keluarga dari nasabah, *CreditScore* adalah skor kredit dari nasabah, *Geography* menunjukkan geografi, *Gender* menunjukkan jenis kelamin, *Age* menunjukkan umur, *Tenure* menunjukkan waktu nasabah telah memiliki akun, *Balance* adalah jumlah uang tersimpan dalam akun nasabah, *NumOfProducts* mengacu pada jumlah produk atau layanan perbankan yang dimiliki oleh seorang nasabah, *isActiveMember* mengacu pada status keanggotaan aktif, dan *Estimated Salary* adalah gaji estimasi dari nasabah.

Dalam dataset ini, target kelas yang digunakan adalah *'Exited'*. Pada data *training*, kelas ini memiliki nilai biner, yaitu 1 dan 0 seperti pada "Fig. 3". Prediksi akan dilakukan terhadap kelas ini dan kalkulasi yang dilakukan adalah mencari probabilitas dari seorang nasabah akan menutup akun menggunakan nilai kontinu. Semakin tinggi nilai nya, maka semakin tinggi kemungkinan orang tersebut akan menutup akun, berlaku sebaliknya untuk nilai probabilitas yang semakin rendah. Melalui data latih, dataset menunjukkan persentase dominan terhadap *Not Exited*. Melalui *heatmap* korelasi fitur kontinu "Fig. 5", tidak ada fitur yang memiliki korelasi yang tinggi. Berdasarkan pengamatan pada *heatmap* "Fig. 4", tidak ada *missing value* ditemukan pada data latih maupun data uji.

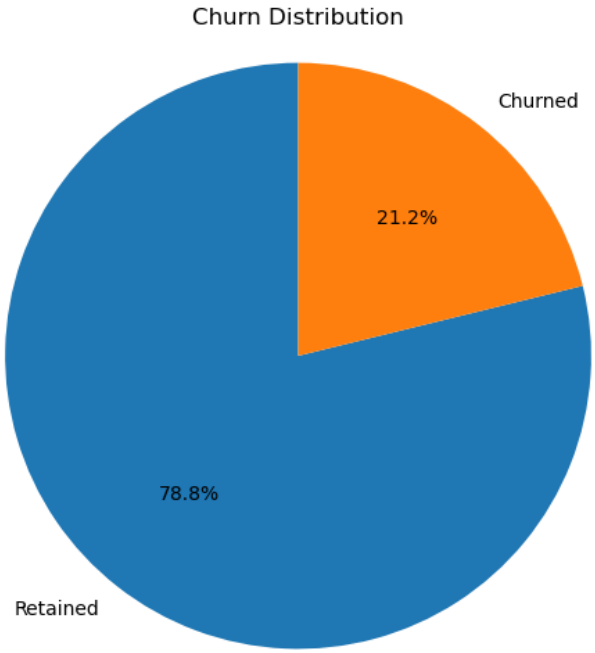

Fig. 3. Distribusi kelas Bank Churn

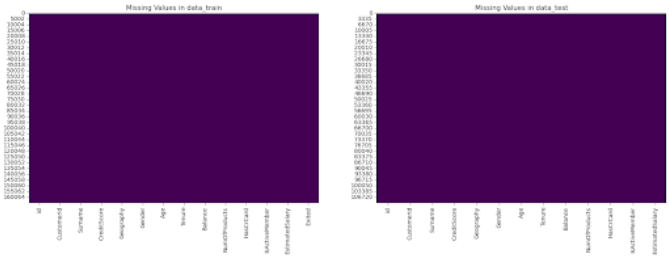

Fig. 4. Nilai NaN Dataset Bank Churn

C. Horse Health Dataset

Dataset Predict Health Outcomes of Horses adalah kumpulan data yang memuat informasi tentang kondisi kesehatan kuda dan keterkaitannya dengan hasil akhir kesehatan mereka, seperti apakah mereka akan sembuh, mati, atau di-euthanasia. Tujuan dari dataset ini adalah untuk memprediksi hasil kesehatan kuda berdasarkan berbagai fitur medis yang tersedia.

Fitur-fitur dataset:

- 'surgery': Apakah kuda tersebut menjalani operasi (yes/no)
- 'age': Usia kuda (adult)
- 'hospital number': Nomor rumah sakit tempat kuda dirawat
- 'rectal temp': Suhu rektal kuda dalam derajat Celcius
- 'pulse': Denyut nadi kuda per menit
- 'respiratory rate': Laju pernapasan kuda per menit

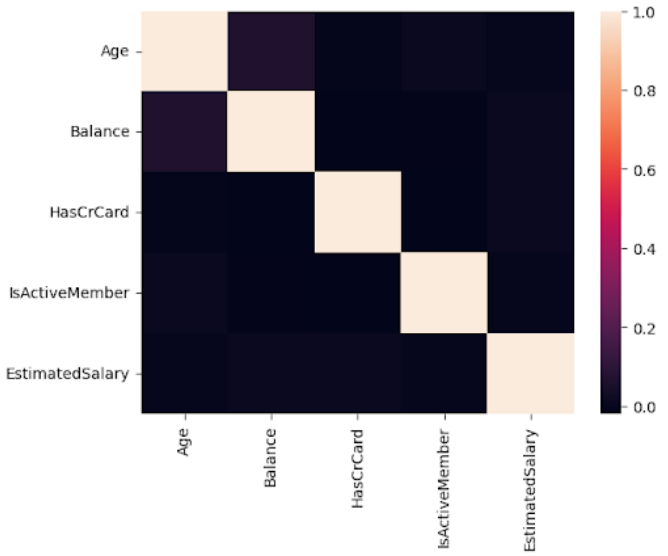

Fig. 5. Heatmap Fitur Dataset Bank Churn

- 'temp of extremities': Suhu ekstrimitas ('cool'/'normal'/'code')
- 'peripheral pulse': Denyut nadi perifer ('normal'/'reduced')
- 'mucous membrane': Warna membran mukosa ('pale pink'/'normal pink'/'dark cyanotic')
- 'capillary refill time': Waktu pengisian kembali kapiler ('less 3 sec'/'more 3 sec')
- 'pain': Tingkat rasa sakit ('alert'/'depressed'/'mild pain'/'severe pain'/extreme pain')
- 'peristalsis': Gerakan peristaltik usus ('normal'/'hypomotile'/'hypermotile'/'absent')
- 'abdominal distention': Distensi abdominal (slight/moderate/none)
- 'nasogastric tube': Kehadiran selang nasogastrik (yes/no)
- 'nasogastric reflux': Refluks nasogastrik (yes/no)
- 'nasogastric reflux pH': pH refluks nasogastrik
- 'rectal exam feces': Hasil pemeriksaan rektal (normal/decreased)
- 'abdomen': Kondisi perut (normal/firm/distend small/distend large)
- 'packed cell volume': Volume sel darah terpak (dalam persen)
- 'total protein': Total protein dalam darah (dalam g/dL)
- 'abdomo appearance': Penampakan cairan perut (serosanguious/cloudy)
- 'abdomo protein': Protein dalam cairan perut (dalam g/dL)
- 'surgical lession': Protein dalam cairan perut (dalam g/dL)
- 'lession 1', 'lession 2', 'lession 3': Identifikasi lesi (numerik)
- 'cp data': Data terkair CP (yes/no)
- 'outcome': Hasil akhir kesehatan kuda (lived/died/euthanized)

Statistik dan distribusi fitur dataset Horse Health:

- Fitur kategorikal seperti *surgery, age, temp of extremities, peripheral pulse, mucousmembrane, capillary refill time, pain, peristalsis, abdominaldistention, nasogastrictube, nasogastricreflux, rectalexamfeces, abdomen, abdomo appearance, surgical lesion, cp data*, dan *outcome* dikodekan menggunakan LabelEncoder.
- Fitur numerik meliputi *hospital number, rectal temp, pulse, respiratory rate, nasogastric reflux pH, packed cell volume, total protein, abdomo protein, lesion 1, lesion 2*, dan *lesion 3*.

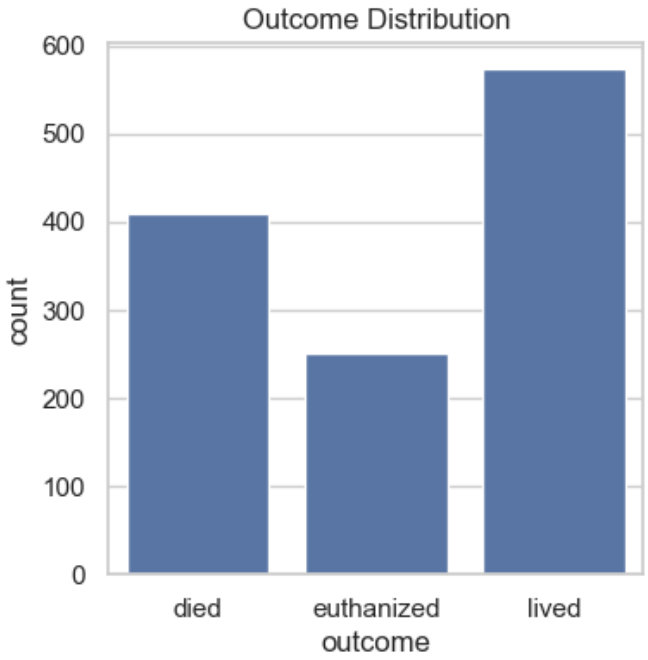

Fig. 6. Distribusi Outcome Dataset Horse Health

Dataset ini menunjukkan distribusi hasil kesehatan kuda sebagai berikut: *lived* (kuda yang bertahan hidup), *died* (kuda yang meninggal), dan *euthanized* (kuda yang di-euthanasia). Grafik distribusi hasil kesehatan "Fig. 6" menunjukkan bahwa sebagian besar kuda dalam dataset ini bertahan hidup, dengan proporsi yang lebih kecil meninggal atau di-euthanasia. Analisis korelasi antar fitur "Fig. 8" dalam dataset ini menunjukkan hubungan antara berbagai fitur dan hasil kesehatan. Heatmap korelasi menunjukkan bahwa beberapa fitur memiliki korelasi yang lebih tinggi dengan hasil kesehatan kuda, yang dapat digunakan untuk membangun model prediktif yang lebih akurat.

D. Keep-it-Dry Dataset

Dataset Keep-it-Dry merepresentasikan hasil dari product testing untuk produk absorben berdasarkan hasil pengukuran dan uji laboratorium tertentu. Setiap produk absorben yang diujikan memiliki *'product code'* dengan empat buah atribut tetap (*fixated attributes*), yang disimpan pada fitur *'attribute 0'* hingga *'attribute 3'*. Tiap *record* pada dataset ini juga

	dtypes	missing#	missing%	uniques	count
surgery	object	0	0.000000	2	1235
age	object	0	0.000000	2	1235
hospital_number	int64	0	0.000000	255	1235
rectal_temp	float64	0	0.000000	43	1235
pulse	float64	0	0.000000	50	1235
respiratory_rate	float64	0	0.000000	37	1235
temp_of_extremities	object	39	0.031579	4	1196
peripheral_pulse	object	60	0.048583	4	1175
mucous_membrane	object	21	0.017004	6	1214
capillary_refill_time	object	6	0.004858	3	1229
pain	object	44	0.035628	6	1191
peristalsis	object	20	0.016194	5	1215
abdominal_distention	object	23	0.018623	4	1212
nasogastric_tube	object	80	0.064777	3	1155
nasogastric_reflux	object	21	0.017004	4	1214
nasogastric_reflux_ph	float64	0	0.000000	26	1235
rectal_exam_feces	object	190	0.153846	5	1045
abdomen	object	213	0.172470	5	1022
packed_cell_volume	float64	0	0.000000	49	1235
total_protein	float64	0	0.000000	83	1235
abdomo_appearance	object	48	0.038866	3	1187
abdomo_protein	float64	0	0.000000	54	1235
surgical_lesion	object	0	0.000000	2	1235
lesion_1	int64	0	0.000000	57	1235
lesion_2	int64	0	0.000000	4	1235
lesion_3	int64	0	0.000000	2	1235
cp_data	object	0	0.000000	2	1235
outcome	object	0	0.000000	3	1235

Fig. 7. Statistik Deskriptif Dataset Horse Health

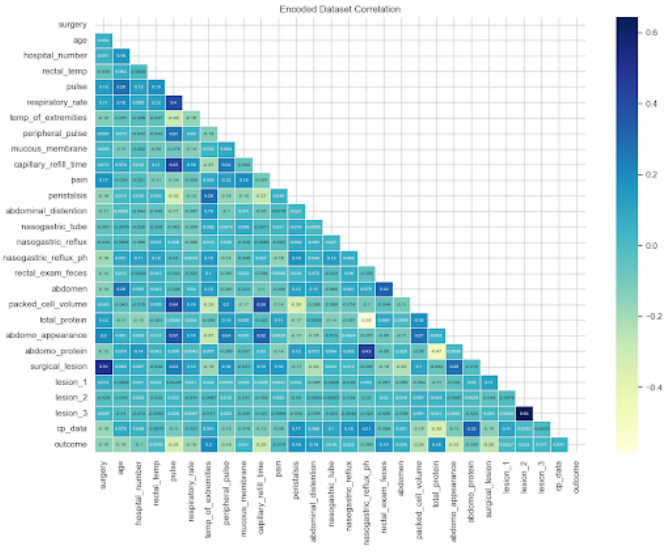

Fig. 8. Korelasi Fitur Dataset Horse Health

memiliki fitur *'loading'*, yang menyatakan banyaknya cairan yang berhasil diserap untuk produk absorben yang diujikan. Hasil pengukuran untuk tiap pengujian berbeda juga tersedia dan direpresentasikan dalam fitur *'measurement 0'* hingga *'measurement 17'*. Kelas target dalam dataset ini adalah *'failure'* dengan nilai biner 0 dan 1, yang menyatakan apakah produk tersebut lolos ataupun gagal.

Training set pada dataset ini memiliki 26,570 *records* dengan 20,775 *records* untuk proses *submission* dengan kelas *'failure'*=0 menjadi *majority class* "Fig. 9". Dari 25 fitur yang tersedia, terdapat 3 fitur yang memiliki tipe data *object*: *'product code'*, *'attribute 0'* dan *'attribute 1'*. Melalui peta persebaran nilai NaN "Fig. 10", terdapat 9 fitur tanpa nilai NaN: *'id'*, *'product code'*, *'loading'*, *'attribute 0'* hingga 'attribute 3', serta *'measurement 0'* hingga *'measurement 2'*. Melalui *heatmap* korelasi fitur kontinu "Fig. 11", tidak ditemukan adanya fitur dengan korelasi linear yang tinggi.

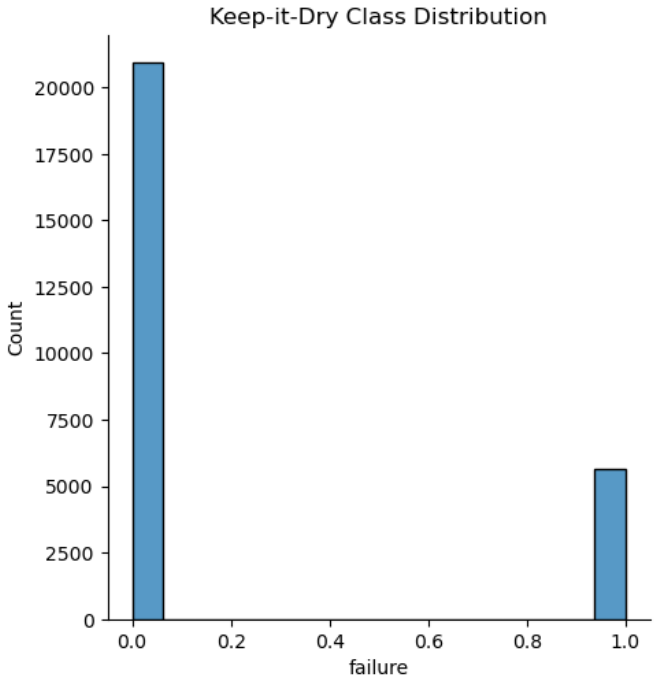

Fig. 9. Distribusi Kelas Dataset Keep-it-Dry

IV. METODOLOGI

A. Imputasi Data

Imputasi data adalah proses mengganti nilai yang hilang dalam dataset dengan nilai yang diperkirakan berdasarkan informasi yang tersedia. Kehilangan data dapat disebabkan oleh berbagai alasan seperti kesalahan pengukuran, pengabaian, atau masalah teknis. Nilai yang hilang dalam dataset bisa mengurangi kualitas analisis dan menyebabkan model *machine learning* menjadi bias jika tidak ditangani dengan benar [35]. Oleh karena itu, imputasi merupakan langkah penting dalam data preprocessing untuk memastikan bahwa

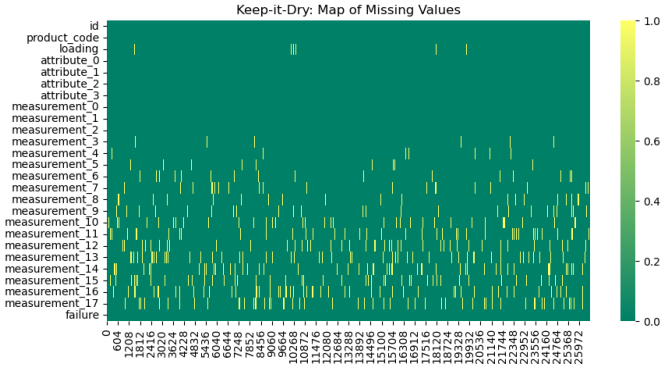

Fig. 10. Distribusi nilai NaN Dataset Keep-it-Dry

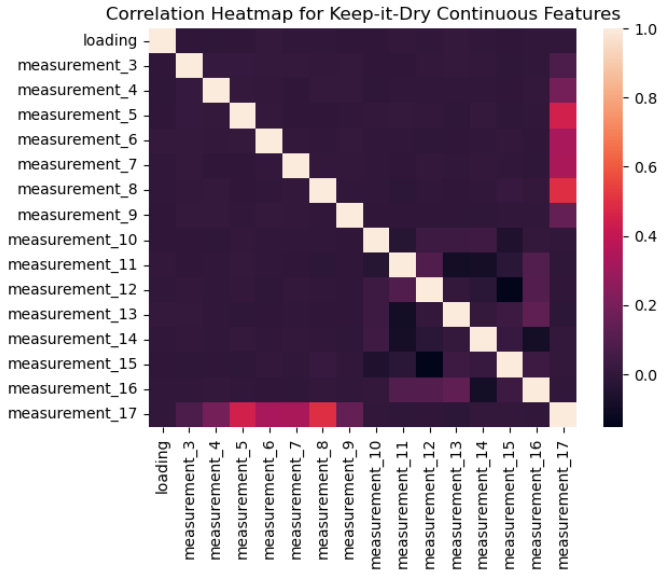

Fig. 11. Korelasi Fitur Dataset Keep-it-Dry

dataset yang digunakan dalam analisis atau pelatihan model adalah representatif dan lengkap.

Dalam penelitian ini, beberapa teknik imputasi data yang digunakan adalah sebagai berikut.

- *Mean Imputation*
  Mengganti nilai yang hilang dengan rata-rata nilai dalam kolom tersebut. Teknik ini sederhana dan sering digunakan, terutama jika data memiliki distribusi normal. Namun, *mean imputation* bisa memperkecil variasi data dan mengabaikan korelasi antar variabel.
- *Median Imputation*
  Mengganti nilai yang hilang dengan nilai tengah dalam kolom tersebut. Opsi ini lebih tahan terhadap *outlier* dibandingkan *mean imputation*, sehingga lebih baik digunakan jika data memiliki distribusi yang *skewed*. *Median imputation* juga mempertahankan distribusi data lebih baik daripada *mean imputation*.
- *Mode Imputation*
  Mengganti nilai yang hilang dengan nilai yang paling sering muncul dalam kolom tersebut. Cocok untuk data kategori, karena menggantikan nilai yang hilang dengan kategori yang paling umum. *Mode imputation* dapat mempertahankan distribusi kategori dalam data, tetapi mungkin tidak ideal untuk data numerik.
- *K-Nearest Neighbors Imputation (KNN Imputation)*
  Menggunakan nilai-nilai dari data yang paling mirip untuk memperkirakan nilai yang hilang. KNN *imputation* mempertimbangkan kedekatan atau kemiripan antara sampel, sehingga dapat menghasilkan imputasi yang lebih akurat dibandingkan metode sederhana seperti mean atau median. Namun, KNN *imputation* bisa jadi komputasi intensif, terutama untuk dataset yang besar.
- *Multiple Imputation* Menghasilkan beberapa set nilai yang hilang untuk mencerminkan ketidakpastian yang lebih baik dalam imputasi. Teknik ini menciptakan beberapa dataset lengkap, menganalisis masing-masing, dan menggabungkan hasilnya untuk memberikan estimasi yang lebih akurat dan tidak bias. *Multiple imputation* menangani ketidakpastian dalam data yang hilang dengan lebih baik dibandingkan metode imputasi tunggal.

B. Data Encoding

Data encoding adalah proses mengubah data kategori menjadi bentuk numerik agar dapat diproses oleh algoritma *machine learning* yang umumnya hanya dapat bekerja dengan data numerik [36]. Data kategori, seperti jenis kelamin atau status perkawinan, perlu diubah menjadi format yang dapat dipahami oleh model matematis. Tanpa *encoding*, algoritma *machine learning* tidak dapat menggunakan informasi dari fitur kategori, karena mereka mengharapkan input dalam bentuk numerik. *Encoding* sangat penting untuk memastikan bahwa model dapat memahami dan memanfaatkan data kategori dengan benar.

Dalam penelitian ini, beberapa teknik *encoding* yang digunakan adalah sebagai berikut.

- *Label Encoding*: memberikan label numerik untuk setiap kategori dalam fitur. Digunakan saat kategori memiliki urutan tertentu (ordinal). Meskipun sederhana, label encoding dapat menimbulkan masalah jika kategori tidak memiliki hubungan ordinal, karena model mungkin akan menafsirkan urutan numerik sebagai representasi hubungan ordinal.
- *One-Hot Encoding*: membuat kolom biner terpisah untuk setiap kategori unik dalam fitur. Menghindari memberikan urutan yang tidak diinginkan pada data kategori. *One-hot encoding* sangat efektif untuk menghindari masalah yang ditimbulkan oleh *label encoding* pada data non-ordinal. Namun, metode ini dapat menyebabkan peningkatan dimensi data yang signifikan jika jumlah kategori banyak.
- *Ordinal Encoding*: mengurutkan kategori dalam fitur dan memberikan nilai numerik berdasarkan urutan tersebut. Cocok untuk data ordinal dimana ada urutan yang jelas di antara kategori, seperti *rating* (buruk, cukup, baik, sangat baik).

- *Binary Encoding*: kombinasi dari *label encoding* dan *one-hot encoding* yang menghasilkan representasi biner. Teknik ini mengurangi dimensi data dibandingkan dengan *one-hot encoding* dan masih memberikan representasi yang baik untuk data kategori.

C. Data Scaling

Data scaling adalah proses menormalkan atau mengubah skala data agar semua fitur berada dalam rentang yang sama, biasanya 0 hingga 1 atau -1 hingga 1. Hal ini penting agar algoritma *machine learning* tidak bias terhadap fitur dengan skala yang lebih besar [37]. Misalnya, fitur yang diukur dalam ribuan unit dapat mendominasi fitur lain yang diukur dalam skala yang lebih kecil, sehingga mempengaruhi kinerja model. *Data scaling* membantu dalam mempercepat konvergensi model selama pelatihan dan meningkatkan performa prediksi.

Dalam penelitian ini, beberapa teknik *data scaling* yang digunakan adalah sebagai berikut.

- *Standard Scaler*: adalah menormalkan data sehingga memiliki mean 0 dan standar deviasi 1. Cocok untuk data yang mendekati distribusi normal. Dengan *Standard Scaler*, kita memastikan bahwa setiap fitur memiliki kontribusi yang seimbang dalam model. *Standard Scaler* menjaga hubungan linear antara fitur dan memungkinkan model untuk lebih efektif mempelajari pola dalam data.
- *MinMax Scaler*: adalah mengubah skala data sehingga berada dalam rentang antara 0 dan 1. Cocok untuk data yang tidak memiliki *outlier* ekstrem. *MinMax Scaler* mempertahankan hubungan proporsional antara nilai asli dan nilai skala.
- *Robust Scaler*: mengurangi pengaruh *outlier* dengan menggunakan median dan *interquartile range*. Cocok untuk data dengan *outlier* yang signifikan. *Robust Scaler* memastikan bahwa *scaling* tidak dipengaruhi oleh nilai ekstrim yang dapat mendistorsi distribusi data. *Robust Scaler* menjaga robustness model terhadap *outlier*.

D. Imbalance Handling

Imbalance handling adalah proses mengatasi ketidakseimbangan kelas dalam dataset, dimana jumlah sampel pada satu kelas jauh lebih banyak dibandingkan kelas lainnya. Ketidakseimbangan ini dapat menyebabkan model *machine learning* bias terhadap kelas mayoritas dan mengabaikan kelas minoritas [38], sehingga menghasilkan prediksi yang tidak akurat. Ketidakseimbangan kelas dapat mengurangi kemampuan model untuk mendeteksi kelas minoritas yang penting. Opsi yang tersedia untuk *imbalance handling* yang digunakan pada penelitian ini adalah sebagai berikut.

- *Oversampling*: menambah jumlah sampel dari kelas minoritas. Teknik ini termasuk ROS (*Random Over Sampler*) maupun teknik seperti SMOTE (*Synthetic Minority Over-sampling Technique*), yang menciptakan sampel baru dari kelas minoritas dengan menginterpolasi antara sampel yang ada. *Oversampling* dapat meningkatkan representasi kelas minoritas dalam dataset. Namun, *oversampling* dapat menyebabkan *overfitting* jika sampel yang baru diciptakan terlalu mirip dengan yang sudah ada.
- *Undersampling*: mengurangi jumlah sampel dari kelas mayoritas, seperti dengan menggunakan RUS (*Random Under Sampler*). Teknik ini menghapus sebagian sampel dari kelas mayoritas untuk mencapai keseimbangan. Meskipun dapat membantu mengatasi ketidakseimbangan, *undersampling* berisiko menghilangkan informasi penting dari kelas mayoritas. *Undersampling* membantu dalam mengurangi ukuran dataset dan mempercepat proses pelatihan.
- *Hybrid Methods*: menggabungkan *oversampling* dan *undersampling* untuk mendapatkan keseimbangan yang lebih baik. *Hybrid methods* memanfaatkan kelebihan dari kedua pendekatan untuk menciptakan dataset yang lebih seimbang. *Hybrid methods* dapat mengurangi risiko overfitting dan underfitting yang mungkin terjadi pada *oversampling* atau *undersampling* tunggal.

E. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) adalah teknik pengurangan dimensi yang mengubah data ke dalam set baru dari variabel (*principal components*) yang mempertahankan variabilitas maksimum dalam data. PCA membantu dalam mengurangi jumlah fitur dan mengatasi masalah multikolinearitas. Dalam PCA, data diubah menjadi komponen utama yang merupakan kombinasi linear dari fitur asli dengan varian terbesar. Komponen utama ini mempertahankan sebanyak mungkin informasi dari data asli. PCA mengidentifikasi arah variabilitas maksimum dalam data dan mentransformasikannya menjadi set baru dari fitur yang tidak berkorelasi.

Tujuan dari PCA adalah mengurangi jumlah fitur dalam dataset, mengatasi masalah multikolinearitas, meningkatkan efisiensi komputasi, dan meningkatkan performa model *machine learning* [39]. Dengan mengurangi dimensi data, PCA membantu dalam visualisasi data, mempercepat waktu pelatihan model, dan mengurangi risiko *overfitting*. PCA juga membantu dalam mengidentifikasi fitur yang paling penting dan mengabaikan *noise* dalam data. Transformasi data menggunakan PCA memungkinkan model untuk fokus pada komponen yang paling signifikan dan mengurangi beban komputasi.

F. Optimalisasi Hyperparameter: GridSearchCV

Akurasi dari suatu model dapat ditingkatkan dengan memanfaatkan GridSearchCV [40]. GridSearchCV adalah sebuah kelas yang disediakan oleh *framework* scikit-learning. Tujuan dari GridSearchCV adalah untuk menentukan kombinasi *hyperparameter* yang menghasilkan nilai terbaik [41]. Grid search bekerja dengan cara melakukan seleksi terhadap model dan *hyperparameter* terbaik. Untuk memperoleh model dan *hyperparameter* terbaik, teknik ini akan melakukan pengujian terhadap seluruh kombinasi *hyperparameter* dari *dictionary* yang telah didefinisikan. Untuk setiap kombinasi yang berhasil dibangkitkan, akan dilakukan validasi terhadap model klasifikasi dengan menggunakan *k-fold cross-validation* (CV). Setiap *k-fold cross-validation* dari model akan menghasilkan

evaluation index yang akan menjadi acuan dalam menentukan model klasifikasi terbaik [42].

$$\mathrm{E} = \frac{1}{K}\sum_{k=1}^{K} E_k \tag{35}$$

Setiap tahapan *K-fold cross-validation* dilakukan dengan melakukan repetisi sebanyak K kali pada data yang terbagi menjadi K bagian. Pada setiap perulangan 1/k dataset digunakan sebagai data uji dan sisanya akan bertindak sebagai data latih [43]. *K-fold cross-validation* akan menghitung rata-rata dari kalkulasi yang telah dilakukan dan menjadikannya sebagai index performansi dari model klasifikasi. Setelah itu, parameter klasifikasi diubah ke kombinasi lainnya dan proses validasi dilakukan kembali. Tahapan ini akan terus berlanjut dan setiap iterasi akan menghasilkan akurasi model. Persamaan dari *K-fold cross-validation* dapat dilihat pada "(35)". Diagram alir dari proses GridSearchCV dapat dilihat pada "Fig. 12".

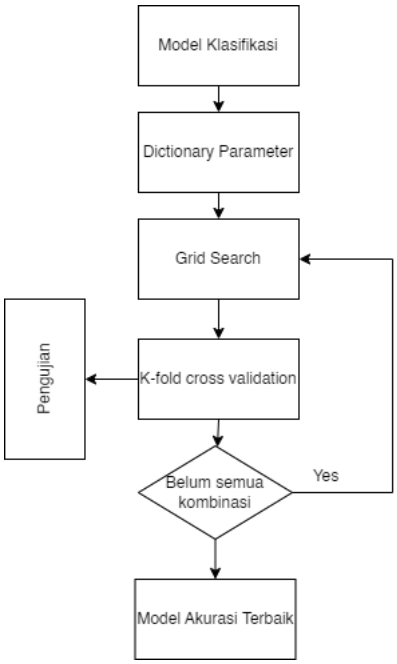

Fig. 12. Diagram Alir GridSearchCV

G. Optimalisasi Hyperparameter: Optuna

Optuna adalah sebuah pustaka yang melakukan optimasi terhadap *hyperparameter tuning*. Untuk menemukan *hyperparameter* terbaik terkadang, melakukan proses training dan evaluasi secara repetitif hanya akan membuang waktu dan tenaga [44]. Oleh karena itu, Optuna berperan untuk melakukan automasi terhadap optimasi *hyperparameter*.

Optuna menggunakan landasan utama seperti fungsi objektif yang digunakan sebagai penentu metrik yang ingin dioptimalkan, ruang pencarian yang mendefinisikan rentang valid atau kumpulan nilai diskrit dari setiap parameter, trial yang mengevaluasi fungsi objektif dengan *hyperparameter* tertentu, dan sampler yang menentukan parameter baru yang digunakan.

Langkah-langkah Optuna dalam memilih *hyperparameter* yang optimal terdiri dari: Penentuan fungsi objektif, pemilihan sampler, Iterasi dan Optimasi [45]. Prosedur dasar Optuna dapat dilihat pada "Fig. 13".

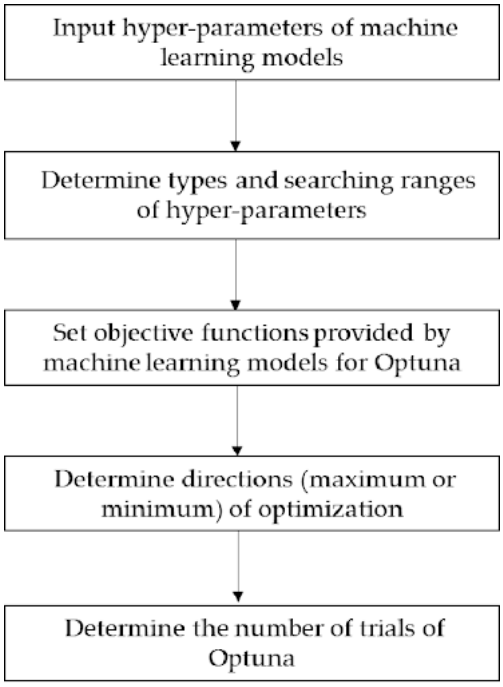

Fig. 13. Diagram Alir Optuna

H. Evaluasi Model

Pada pengujian yang dilakukan di artikel ini, evaluasi model dilakukan dengan menggunakan metode tabulasi silang (*Confusion Matrix*). Proses evaluasi ini adalah dengan menyusun matriks yang berisi kelas data asli pada barisnya dan kelas data hasil prediksi pada kolomnya [46]. Ilustrasi *Confusion Matrix* dapat dilihat pada "Fig. 14".

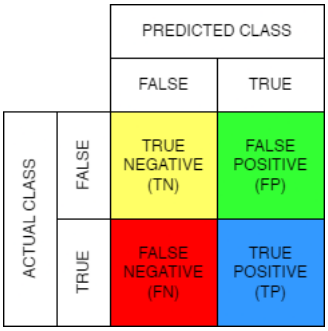

Fig. 14. Confusion Matrix

Pada "Fig. 14", terdapat 4 lokasi grid dengan kondisi yang berbeda. *True Negative* menandakan jumlah prediksi negatif yang berhasil diprediksi secara benar. *False Positive* menandakan jumlah *actual negative* yang salah diprediksi sebagai positif. *False Negative* menandakan jumlah *actual positive* yang salah diprediksi sebagai *negative* [47].

Selain itu, digunakan beberapa metode *scoring* sebagai berikut.

- **Recall**
  Recall (perolehan) berhubungan dengan kemampuan suatu sistem temu balik dalam menemukan dokumen yang relevan. Hal ini berarti perolehan (*recall*) adalah bagian dari proses temu balik informasi yang dapat digunakan sebagai alat ukur tingkat efektivitas suatu sistem temu balik informasi [48]. Rumus *Recall* dapat dilihat pada "(36)".
- **Precision**
  Precision adalah jumlah kelompok dokumen relevan dari total jumlah dokumen yang ditemukan oleh sistem. Berdasarkan penjelasan tersebut, rumus *precision* dapat dilihat pada "(37)".
- **Accuracy**
  Accuracy adalah kedekatan antara nilai prediksi dengan nilai aktual. Dalam memilih *accuracy*, seluruh komponen terlibat. Rumus *accuracy* dapat dilihat pada "(38)".
- F-Measure (F1-Score)
  Metrik ini menggabungkan precision dan recall, memberikan gambaran menyeluruh tentang kemampuan model dalam mengidentifikasi contoh positif secara akurat dan meminimalkan false positive. Rumus F-Measure dapat dilihat pada "(39)".
- **AUC-ROC**
  AUC-ROC adalah metrik evaluasi yang mengukur kemampuan model klasifikasi dalam membedakan antara kelas positif dan negatif. Metrik ini dihitung dengan menghitung luas area di bawah kurva ROC (*Receiver Operating Characteristic*). Kurva ROC diplotkan dengan *True Positive Rate* (TPR) pada sumbu-y dan *False Positive Rate* (FPR) pada sumbu-x. AUC = 1.0 berarti Performa model sempurna, semua contoh positif diidentifikasi dengan benar tanpa false positive. AUC = 0.5 berarti performa model tidak lebih baik dari menebak secara acak. Nilai antara 0.5 dan 1.0 berarti menunjukkan performa model di antara acak dan sempurna [48].
  AUC tidak memberikan informasi tentang presisi dan recall secara langsung, sehingga perlu dipertimbangkan metrik lain untuk mengukur performa model secara komprehensif. AUC dapat sensitif terhadap outlier dalam data, sehingga perlu dilakukan outlier detection sebelum menghitung AUC [49].
- **Cross-Validation (CV)**
  Cross-Validation (CV) adalah teknik evaluasi model yang membagi data menjadi beberapa subset (fold) dan melatih model beberapa kali dengan menggunakan subset yang berbeda sebagai data latih dan data uji. Tujuannya adalah untuk mendapatkan perkiraan performa model yang lebih akurat dan stabil. *K-fold Cross-Validation* membagi data menjadi k subset, dan model dilatih k kali dengan menggunakan k-1 subset sebagai data latih dan 1 subset sebagai data uji. *Leave-one-out Cross-Validation* membagi data menjadi n subset, di mana n adalah jumlah data. Model dilatih n kali dengan menggunakan n-1 subset sebagai data latih dan 1 data sebagai data uji. *Stratified Cross-Validation* membagi data menjadi subset berdasarkan kelasnya, dan CV dilakukan dengan memastikan proporsi kelas pada data latih dan data uji sama dengan proporsi kelas pada data keseluruhan. [50]. CV memiliki kelebihan dapat digunakan untuk memperkirakan performa model yang lebih akurat dan stabil, memilih model terbaik, serta membantu dalam proses pemilihan parameter terbaik. Kekurangan dari CV adalah waktu komputasi yang cukup lama.

$$Recall = \frac{TP}{TP + FN} \tag{36}$$

$$Precision = \frac{TP}{TP + FP} \tag{37}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{38}$$

$$F1 = 2.\frac{Precision * Recall}{Precision \times Recall} \tag{39}$$

V. Hasil dan Diskusi

A. Dataset Spaceship

Pada tahap pemrosesan awal, yang dilakukan adalah menghapus kolom yang tidak relevan untuk mengurangi kompleksitas data, kemudian melakukan one-hot encoding dan label encoding pada kolom kategorikal. Kolom boolean diubah menjadi tipe data integer untuk memastikan konsistensi format data. Selanjutnya, melakukan normalisasi Z-Score pada variabel kontinu untuk menstandarkan data sehingga setiap fitur memiliki mean nol dan standar deviasi satu, memastikan tidak ada fitur yang mendominasi karena skala yang berbeda. Scaling data dilakukan dengan menggunakan Standard Scaling (ST). dari tahapan tersebut terbentuk 2 clean dataset dengan teknik pemrosesan awal yang yang akan diujikan pada tahap dataset selection untuk masing masing model.

Pada percobaan menggunakan model klasifikasi KNN, diperoleh akurasi sebesar 0.52 dengan menggunakan PCA (Component Analysis) dan k=3. Namun ketika menambahkan beberapa parameter (Knn__n_neighbors, Knn__weights, knn__metric) dengan gridsearchCV dan menggunakan StratifiedKFold dengan 10 fold menghasilkan skor sebesar 0.78. Ini memperlihatkan bahwa gabungan dari gridsearchCV dan stratifiedKFold menghasilkan kombinasi yang baik dalam meningkatkan performa model.

Percobaan dengan model linear SVC, diperoleh skor sebesar 0.50 dengan menggunakan Z-score normalization dan parameter c = 47. Hasil ini menunjukkan bahwa model SVM tersebut kurang cocok dalam dataset ini.

Pada percobaan dengan model LightGBM Classifier (Gradient Boosting Machine) menggunakan standard scaler untuk scaling, hyperparameter tuning dengan parameter n_estimators, max_depth, learning_rate, subsample, colsample_bytree pada gridsearchCV, dan menggunakan metode

cross-validation menghasilkan kombinasi yang baik dan menghasilkan skor 0.79541. Namun ketika menggunakan StratifiedKFold dengan 10 fold dalam gridsearchCV menghasilkan performa model yang lebih baik dan menghasilkan cross-validation score 0.81, accuracy on test 0.799 dan skor ketika disubmit pada kaggle sebesar 0.80640 ini merupakan skor tertinggi dari semua model untuk dataset ini.

Pada percobaan dengan ensemble model menggunakan stacking classifier dengan algoritma XGBClassifier, CatBoostClassifier, LGBMClassifier sebagai base modelnnya, dan logistic regression sebagai final modelnya. Standarisasi data dengan standard scaler dan StratifiedKFold dengan 10 fold untuk menangani data yang tidak seimbang pada cross validasi. Kemudian optimasi parameter dilakukan menggunakan gridsearchCV dengan berbagai kombinasi parameter dari base model dan meta model digunakan untuk mencari konfigurasi yang memberikan akurasi terbaik, dan memberikan nilai akurasi 0.801 dan skor 0.797 pada Kaggle. Tidak lebih baik dari model lightgbm sendiri, mungkin terdapat overfitting pada model ensemble ini.

SVM	0.50596
KNN	0.77180
XGBoost	0.79682
Ensemble (Stacking)	0.79705
CatBoost	0.79752
LightGBM	0.80640

Fig. 15. Log Submission Dataset Spaceship

Pada "Fig. 15", dapat dilakukan inferensi jika model klasifikasi berbasis tree seperti LGBMClassifier, CatBoostClassifier, XGB memiliki performa yang lebih baik. Model-model tersebut diperlakukan sama dengan standard scaler dan menggunakan gridsearch untuk hyperparameter tuning dan StratifiedKFold, yang membedakan hanya pada bagian classifiernya saja. Untuk algoritma KNNClassifier, proses training dengan menggunakan PCA cenderung memperoleh hasil yang lebih baik. Dalam klasifikasi dataset Spaceship Titanic, model LightGBM dengan dengan data hasil pemrosesan awal menggunakan standard scaler memberikan performa paling optimal jika dibandingkan dengan algoritma klasifikasi lainnya.

B. Dataset Bank Churn

Pemrosesan awal dataset Bank Churn dilakukan dengan melakukan tiga tahapan utama: scaling, encoding, dan feature engineering. Tahapan imputasi tidak dilakukan karena dataset tidak memiliki missing value. Scaling data dilakukan dengan menggunakan standard scaling (ST). Encoding pada dataset menggunakan one-hot encoding pada fitur Geography dan Gender. Tahap akhir, dilakukan proses feature engineering pada dataset surname. Dari proses tersebut, didapatkan sebuah clean dataset dengan tipe data numerik yang akan diujikan pada beberapa model.

Pada percobaan menggunakan model klasifikasi Gaussian Naive Bayes (GNBClassifier), diperoleh akurasi sebesar 0.80 dengan menggunakan metode evaluasi cross-validation. Ketika menambahkan Principal Component Analysis (PCA) dengan nilai n_components = 10, diperoleh akurasi sebesar 0.81. Penggunaan PCA memperlihatkan nilai akurasi yang sedikit lebih baik pada model ini.

Pada percobaan menggunakan model Gradient Boost Algorithm (XGBClassifier), dengan nilai PCA yang sama, diperoleh hasil evaluasi 0.860. Ketika n_components = 13, nilai evaluasi meningkat menjadi 0.865. Berdasarkan percobaan ini, dapat disimpulkan bahwa PCA dengan n_components yang lebih sedikit kurang relevan digunakan dalam XGBClassifier. Untuk meningkatkan performa, model ini ditambahkan hyperparameter tuning dengan menggunakan gridsearchCV. Hasilnya nilai evaluasi yang diperoleh adalah sebesar 0.868. Penggunaan hyperparameter yang kurang cocok pada model ini adalah subsample karena dapat berpengaruh pada jumlah sampel dalam pelatihan.

Model KNN menggunakan PCA n_components=0.1. Pada model ini, nilai K yang digunakan adalah 20 dan parameter yang digunakan adalah weights dan metrics. Dengan metric ROC-AUC, diperoleh nilai evaluasi 0.87, sedangkan dengan gridsearch diperoleh akurasi 0.85. Model KNN menunjukkan parameter weight berupa distance dan metrics. Model KNN dengan manhattan distance menghasilkan nilai akurasi yang lebih baik.

LBGMClassifier menghasilkan nilai relatif lebih tinggi dibandingkan dengan model lainnya untuk dataset ini. Model ini menggunakan gridsearch dengan parameter n_estimators, learning_rate, dan max_depth. Penggunaan parameter subsample awalnya digunakan, akan tetapi hasil lebih baik ketika tidak menggunakan sub sample. Evaluasi dilakukan dengan ROC-AUC dan menghasilkan nilai evaluasi 0.90 dan evaluasi menggunakan randomized search diperoleh 0.898, dari hasil gridsearch, f1-score yang diperoleh sebesar 0.87 dan cross-validation menghasilkan skor 0.899.

Penggunaan Model CatBoostClassifier menghasilkan nilai evaluasi yang serupa dengan LGBMClassifier.. Model ini menggunakan GridSearchCV dengan parameter iterations, learning_rate, depth, l2_leaf_reg, border_count, dan loss_function. Evaluasi dilakukan dengan menggunakan ROC-AUC dan cross-validation. Evaluasi menggunakan ROC-AUC menghasilkan skor sebesar 0.90, sedangkan dengan menggunakan cross-validation diperoleh skor sebesar 0.899.

Pada "Fig. 16", dapat dilakukan inferensi jika model klasifikasi berbasis tree seperti LGBMClassifier dan CatBoostClassifier memiliki performa yang lebih baik. Untuk algoritma klasifikasi GaussianNB dan KNNClassifier, proses training dengan menggunakan PCA cenderung memperoleh hasil yang lebih baik. Dalam klasifikasi dataset Bank Churn, model CatBoostClassifier dengan dengan data hasil pemrosesan awal menggunakan min-max scaling (MM) memberikan performa paling optimal jika dibandingkan dengan algoritma klasifikasi

Submission Log (Akurasi Final)

MODEL	AKURASI
GNB	0.79082
XGB	0.88204
KNN	0.85972
LGBM	0.88376
CATBOOST	0.88800

Fig. 16. Log Submission Dataset Bank Churn

lainnya.

C. Dataset Horse Health

Pada tahap preprocessing, yang dilakukan adalah beberapa langkah preprocessing dan feature engineering. Pertama, beberapa kolom kategorikal diencoding menggunakan LabelEncoder dan OneHotEncoder untuk mengubah nilai kategorikal menjadi numerik. Nilai tertentu dalam kolom seperti ”pain”, ”peristalsis”, ”rectal_exam_feces”, dan ”nasogastric_reflux” diganti untuk menyederhanakan kategori. Selanjutnya, nilai yang hilang dalam beberapa kolom diisi dengan nilai default dan di-mapping ke nilai numerik yang sesuai. Kolom ”lesion_3” dihapus karena dianggap tidak diperlukan. Feature engineering dilakukan dengan mengubah kolom ”lesion_2” menjadi biner dan membuat fitur baru ”abs_rectal_temp” sebagai nilai absolut dari selisih antara ”rectal_temp” dan 37.8, kemudian kolom ”rectal_temp” dihapus. Langkah-langkah ini dilakukan untuk memastikan data dalam format yang sesuai untuk digunakan dalam model machine learning, dengan menangani nilai yang hilang, mengubah fitur kategorikal menjadi numerik, dan melakukan features engineering untuk meningkatkan kualitas data.

Setelah tahap preprocessing, setiap algoritma dalam model machine learning mengalami serangkaian langkah yang berbeda untuk mencapai performa terbaiknya. Algoritma LightGBM memanfaatkan teknik SMOTE untuk menangani ketidakseimbangan kelas sebelum melakukan pemodelan, awalnya mencapai akurasi sebesar 0.79. Setelah optimasi hyperparameter menggunakan GridSearch yang menghasilkan parameter terbaik ’learning_rate’: 0.2, ’min_child_samples’: 20, ’n_estimators’: 500, ’num_leaves’: 70, akurasi meningkat menjadi 0.81. CatBoost menggunakan class weights untuk menangani ketidakseimbangan kelas, mencapai akurasi 0.72 tanpa optimasi tambahan. RandomForest juga menggunakan class weights dengan akurasi awal 0.72, tetapi setelah optimasi hyperparameter dengan GridSearchCV, akurasi sedikit naik menjadi 0.73 dengan parameter terbaik ’bootstrap’: True, ’max_depth’: 10, ’min_samples_leaf’: 1, ’min_samples_split’: 2, ’n_estimators’: 100

Sebelum pemodelan dengan SVM, data dinormalisasi menggunakan MinMaxScaler, menghasilkan akurasi awal sebesar 0.49, menunjukkan bahwa normalisasi saja tidak cukup untuk meningkatkan performa model ini. XGBoost, seperti LightGBM, menggunakan SMOTE untuk menangani ketidakseimbangan kelas dan mencapai akurasi awal sebesar 0.77. Setelah optimasi hyperparameter menggunakan GridSearchCV dengan parameter terbaik ’colsample_bytree’: 0.3, ’learning_rate’: 0.2, ’max_depth’: 7, ’n_estimators’: 200, akurasi meningkat menjadi 0.79. Decision Tree menggunakan MinMaxScaler untuk normalisasi dan class weights, mencapai akurasi awal 0.57. Setelah optimasi hyperparameter menggunakan GridSearchCV dengan parameter terbaik ’criterion’: ’gini’, ’max_depth’: None, ’min_samples_leaf’: 10, ’min_samples_split’: 2, akurasi meningkat menjadi 0.64.

K-Nearest Neighbors (KNN) menggunakan MinMaxScaler untuk normalisasi dan SMOTE untuk menangani ketidakseimbangan kelas, mencapai akurasi awal sebesar 0.63. Setelah optimasi hyperparameter menggunakan GridSearchCV dengan parameter terbaik ’metric’: ’manhattan’, ’n_neighbors’: 3, ’weights’: ’distance’, akurasi sedikit meningkat menjadi 0.64. Naive Bayes menggunakan StandardScaler untuk standarisasi data dan SMOTE sebelum pemodelan, menghasilkan akurasi awal sebesar 0.62. Setelah optimasi hyperparameter menggunakan GridSearchCV dengan parameter terbaik ’var_smoothing’: 1e-05, akurasi berada di angka 0.68. Nilai ini dapat dilihat pada “Fig. 17”

Submission Log (Akurasi Final)

MODEL	AKURASI
KNN	0.67682
SVM	0.46951
NaiveBayes	0.70121
Decision Tree	0.67682
LightGBM	0.81707
CatBoost	0.76219
LightGBM Optuna	0.83536
RandomForest	0.76219
XGBoost	0.79878

Fig. 17. Log Submission Dataset Horse Health

D. Dataset Keep-it-Dry

Pemrosesan awal dataset Keep-it-Dry dilakukan dengan melakukan tiga tahapan utama: imputasi, scaling dan encoding, serta imbalance handling. Pada tahapan imputasi, digunakan dua teknik imputasi dengan agregasi data berdasarkan product_code: K-Nearest Neighbour Imputation (KI) dan Iterative Imputation (II). Proses scaling untuk data kontinu dengan menggunakan min-max scaling (MM), standard scaling (ST) dan robust scaling (RO) dan ketiganya menggunakan one-hot

encoding. Proses imbalance handling dilakukan dengan teknik Random Over Sampler (ROS), Random Under Sampler (RUS), SMOTE, SMOTENC, dan ADASYN. Dengan kombinasi terbentuk 16 clean dataset dengan beragam teknik pemrosesan awal yang akan diujikan pada tahap dataset selection untuk masing-masing model. Untuk setiap model yang digunakan, akan dilakukan dataset selection dengan melakukan testing ke-16 clean dataset dengan menggunakan base model dengan penilaian berdasarkan F1-score.

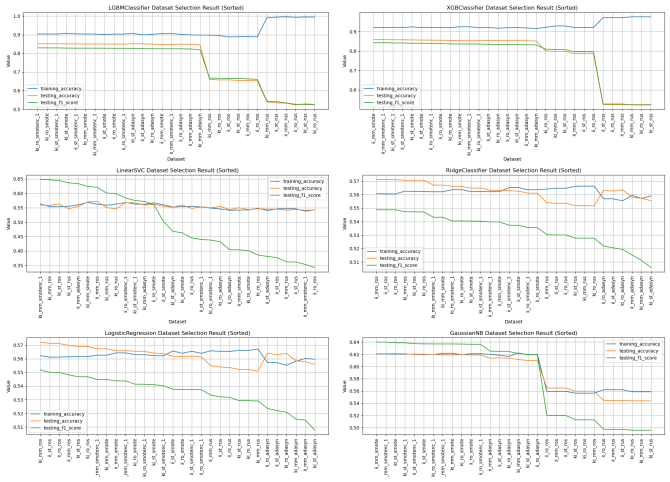

Fig. 18. Dataset Selection untuk Dataset Keep-it-Dry

Berdasarkan hasil dari dataset selection untuk berbagai model "Fig. 18", teknik imbalance handling dengan SMOTE, SMOTENC, dan ADASYN (kode: _*_*smotenc_1, _*_*smote, dan _*_*adasyn) memiliki performa yang lebih baik dalam proses klasifikasi pada training set dengan nilai F1-score yang tinggi jika dibandingkan dengan imbalance handling dengan ROS ataupun RUS (kode: _*_*ros dan _*_*rus). Akan tetapi, model yang dilatih dengan data resampling SMOTE, SMOTENC, dan ADASYN memberikan performa generalisasi yang jauh lebih buruk dari model yang dilatih dengan data resampling ROS dan RUS pada submission dataset. Oleh karenanya, dataset yang digunakan dalam proses hyperparameter tuning dan pelatihan model akan menggunakan dataset hasil resampling ROS atau RUS dengan nilai F1-score terbaik. Terlihat pula jika tree-based classifier seperti LGBMClassifier dan XGBClassifier memiliki rata-rata testing F1-score untuk proses dataset selection yang lebih baik.

Dari hasil data selection, selanjutnya adalah proses training dengan dataset yang menawarkan F1-score tertinggi. Sebagai bahan komparasi lanjutan, dilakukan pula training dengan dataset hasil resample ROS dan RUS. Proses Principal Component Analysis (PCA) diaplikasikan untuk meningkatkan performa generalisasi model. PCA dilakukan dengan menggunakan nilai variabel 'n_components' yang optimal untuk masing-masing base model dengan penilaian F1-score secara iteratif untuk rentang nilai n_components [10, 23]. Proses roses training dijalankan bersamaan dengan proses hyperparameter tuning dengan validation data sebesar 75% dari total training data. Untuk meningkatkan performa generalisasi model pada unseen data, digunakan pula cross-validation dengan folds sebanyak 5. Dengan best estimator yang didapatkan melalui hyperparameter tuning dengan cross-validation ini, dilakukan prediksi data submission.

Submission Log (Akurasi Final) Keep It Dry

MODEL	AKURASI
KNN	0.51718
LinearSVC	0.56586
NaiveBayes	0.55629
LightGBM	0.56553
LogReg	0.56258
AdaBoost	0.56063
RC	0.56142
XGBoost	0.52906

Fig. 19. Log Submission Dataset Keep-it-Dry

Terlihat pada "Fig. 19", hasil akurasi pada data submission aktual menandakan adanya perbedaan antara testing F1-score dengan actual score. Model klasifikasi berbasis tree seperti XGBClassifier dan LGBMClassifier mengindikasikan adanya overfit pada data submission, bahkan setelah dilakukan hyperparameter tuning dengan cross-validation. Melalui retraining dan resubmission menggunakan dataset dengan testing F1-score yang lebih rendah (RUS-resampled), XGBClassifier dan LGBMClassifier justru memberikan hasil akurasi yang lebih baik pada data submission. Di sisi lain, model klasifikasi linear dengan regularisasi L2 seperti RidgeClassifier (RC) dan LogisticRegression (LR) serta kernelized Linear SVM Classifier (LinearSVC), dengan testing F1-score yang lebih buruk dibandingkan XGBClassifier dan LGBMClassifier, mampu memberikan actual score data submission yang lebih baik. Hal yang sama juga berlaku untuk model klasifikasi Gaussian Naive-Bayes (GaussianNB) dan K-Nearest Neighbors (KNNClassifier). Dari hasil submission, terlihat jika model RC, LR, LinearSVC, GaussianNB dan KNNClassifier yang dilatih dengan dataset ROS-resampled memberikan performa aktual yang lebih baik pada data submission jika dibandingkan dengan dataset hasil synthetic resampling (SMOTE/SMOTENC/ADASYN).

Dari "Fig. 19", dapat dilakukan inferensi jika model klasifikasi berbasis tree mampu memberikan performa yang lebih baik dengan melakukan training pada data dengan RUS-resampling dan PCA, membantu mengurangi dampak overfitting pada unseen data. Untuk algoritma klasifikasi RC, LR, LinearSVC, GaussianNB dan KNNClassifier, proses training pada data dengan ROS-resampling dan PCA memberikan per-

forma yang lebih baik secara keseluruhan jika dibandingkan dengan algoritma klasifikasi berbasis tree. Dalam klasifikasi dataset Keep-it-Dry, model LinearSVC dengan hyperparameter 'C': 4.641588833612777, 'loss': 'squared_hinge', 'penalty': 'l2' dengan data hasil pemrosesan awal menggunakan robust scaling (RO) dan ROS-resampling memberikan performa paling optimal jika dibandingkan dengan algoritma klasifikasi lainnya. Dengan teknik pemrosesan awal yang sama, RC dan LR dengan hyperparameter tuning menggunakan Grid Search dan Cross-Validation juga memberikan performa yang baik setelah LinearSVC.

VI. Simpulan

Pada Proyek Akhir ini, kami mengeksplorasi sejumlah teknik untuk menangani ketidakseimbangan kelas dan proses resampling data dengan tujuan meningkatkan kinerja model klasifikasi. Kami mengevaluasi serta membandingkan teknik-teknik seperti Synthetic Minority Over-sampling Technique (SMOTE), SMOTE for Nominal and Continuous features (SMOTENC), Adaptive Synthetic Sampling (ADASYN), Random Over Sampling (ROS), dan Random Under Sampling (RUS). Temuan penelitian menunjukkan bahwa meskipun teknik SMOTE, SMOTENC, dan ADASYN mampu menghasilkan kinerja yang baik pada data pelatihan, namun cenderung mengalami overfitting pada data pengujian. Sebaliknya, teknik ROS dan RUS menunjukkan performa yang lebih konsisten dan unggul pada data pengujian, sehingga lebih kami pilih untuk digunakan dalam tahap tuning hyperparameter dan pelatihan model.

Penerapan Principal Component Analysis (PCA) juga terbukti efektif dalam meningkatkan kinerja generalisasi model dengan mengurangi dimensi data dan mengatasi masalah multikolinearitas. PCA membantu model fokus pada komponen yang paling signifikan, yang pada akhirnya meningkatkan akurasi prediksi. PCA digabungkan dengan berbagai model klasifikasi dan memberikan hasil yang positif, menunjukkan pentingnya teknik ini dalam proses preprocessing data sebelum pelatihan model.

Model klasifikasi yang kami uji meliputi algoritma berbasis pohon keputusan seperti Extreme Gradient Boosting (XGBClassifier) dan Light Gradient Boosting Machine (LGBMClassifier), serta model linear seperti RidgeClassifier, LogisticRegression, dan Linear Support Vector Classification (LinearSVC). Hasil eksperimen menunjukkan bahwa model berbasis pohon keputusan cenderung mengalami overfitting, sementara model linear menunjukkan performa yang lebih stabil dan dapat diandalkan pada data pengujian. Hal ini menekankan pentingnya pemilihan model yang sesuai berdasarkan karakteristik dataset yang digunakan.

Selain itu, kami juga menguji metode ensemble seperti Stacking, yang menghasilkan kinerja yang kompetitif namun rentan terhadap overfitting jika tidak ditangani dengan benar. Pemilihan dan kombinasi model dalam ensemble harus dilakukan dengan hati-hati untuk memastikan generalisasi yang baik pada data pengujian.

Secara keseluruhan, proyek ini menunjukkan bahwa penanganan ketidakseimbangan kelas, penerapan PCA, dan penggunaan metode ensemble dapat signifikan meningkatkan kinerja model klasifikasi. Implementasi teknik resampling yang tepat dan tuning hyperparameter yang cermat adalah langkah penting dalam pengembangan model yang akurat dan robust. Proyek ini memberikan wawasan berharga bagi praktisi dalam mengembangkan model klasifikasi yang efektif.

VII. Kode Program dan Kontribusi

- Kode program yang digunakan dalam studi komparasi algoritma klasifikasi *Supervised Learning* pada artikel ini dapatdiakses pada tautan berikut: https://github.com/rayhanegar/Bntr-y-lg-diskusi
- Laporan kontribusi dan laporan kemajuan selama proses studi komparasi algoritma klasifikasi *Supervised Learning* dapat diakses pada tautan berikut: https://s.ub.ac.id/progresssppmdfinal

References

[1] GeeksforGeeks, "Machine Learning Tutorial," GeeksforGeeks, May 30, 2018. https://www.geeksforgeeks.org/machine-learning.

[2] "Machine Learning," Coursera, 2018. https://www.coursera.org/specializations/machine-learning-introduction.

[3] Machine Learning, "Machine Learning," IEEE DataPort, 2024. https://ieee-dataport.org/taxonomy/term/2005/all?page=2.

[4] Shai Shalev-Shwartz and Shai Ben-David, Understanding machine learning: From foundations to algorithms. Cambridge Etc: Cambridge University Press, 2014.

[5] F. Maymí and S. Lathrop, "AI in Cyberspace: Beyond the Hype," 2024.

[6] V. V. Putri Wibowo, Z. Rustam, and J. Pandelaki, "Classification of Brain Tumor Using K-Nearest Neighbor-Genetic Algorithm and Support Vector Machine-Genetic Algorithm Methods," in 2021 International Conference on Decision Aid Sciences and Application, Sakheer, Bahrain: IEEE, Dec. 2021, pp. 1077–1081. doi: 10.1109/DASA53625.2021.9682341.

[7] N. F. B. M. Noor, H. S. Sipail, N. Ahmad, and N. M. Noor, "Covid-19 Severity Classification Using Supervised Learning Approach," in 2021 IEEE National Biomedical Engineering Conference, Kuala Lumpur, Malaysia: IEEE, Nov. 2021, pp. 151–156. doi: 10.1109/NBEC53282.2021.9618747.

[8] A. P. Pawlovsky and H. Matsuhashi, "The use of a novel genetic algorithm in component selection for a kNN method for breast cancer prognosis," in 2017 Global Medical Engineering Physics Exchanges/Pan American Health Care Exchanges, Tuxtla-Gutierrez, Mexico: IEEE, Mar. 2017, pp. 1–5. doi: 10.1109/GMEPE-PAHCE.2017.7972084.

[9] A. C. Müller and S. Guido, Introduction to Machine Learning with Python: A Guide for Data Scientists. Beijing: O'reilly, 2017.

[10] K. Kim, "Normalized class coherence change-based k NN for classification of imbalanced data," Pattern Recognition, vol. 120, p. 108126, Dec. 2021, doi: 10.1016/j.patcog.2021.108126.

[11] K. Yuk Carrie Lin, "Optimizing variable selection and neighbourhood size in the K-nearest neighbour algorithm," Computers and Industrial Engineering, vol. 191, p. 110142, May 2024, doi: 10.1016/j.cie.2024.110142.

[12] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, "A novel k NN algorithm with data-driven k parameter computation," Pattern Recognition Letters, vol. 109, pp. 44–54, Jul. 2018, doi: 10.1016/j.patrec.2017.09.036.

[13] A. Rokem and K. Kay, "Fractional ridge regression: a fast, interpretable reparameterization of ridge regression," GigaScience, vol. 9, no. 12, p. giaa133, Nov. 2020, doi: 10.1093/gigascience/giaa133.

[14] B. Akturk, U. Beyaztas, H. L. Shang, and A. Mandal, "Robust functional logistic regression," Adv Data Anal Classif, Feb. 2024, doi: 10.1007/s11634-023-00577-z.

[15] M. Arifuzzaman, Md. R. Hasan, T. J. Toma, S. B. Hassan, and A. K. Paul, "An Advanced Decision Tree-Based Deep Neural Network in Nonlinear Data Classification," Technologies, vol. 11, no. 1, p. 24, Feb. 2023, doi: 10.3390/technologies11010024.

[16] F. Aaboub, H. Chamlal, and T. Ouaderhman, "Analysis of the prediction performance of decision tree-based algorithms," in 2023 International Conference on Decision Aid Sciences and Applications (DASA), Annaba, Algeria: IEEE, Sep. 2023, pp. 7–11. doi: 10.1109/DASA59624.2023.10286809.

[17] R. G. Mantovani, T. Horvath, R. Cerri, J. Vanschoren, and A. C. P. L. F. De Carvalho, "Hyper-Parameter Tuning of a Decision Tree Induction Algorithm," in 2016 5th Brazilian Conference on Intelligent Systems (BRACIS), Recife: IEEE, Oct. 2016, pp. 37–42. doi: 10.1109/BRACIS.2016.018.

[18] Y.-C. Zhang and L. Sakhanenko, "The naive Bayes classifier for functional data," Department of Statistics and Probability, Michigan State University, East Lansing, MI, USA, Accepted April 27, 2019.

[19] A. Khajenezhad, M. A. Bashiri, and H. Beigy, "A distributed density estimation algorithm and its application to naive Bayes classification," Sharif Intelligent Systems Laboratory, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, Accepted October 20, 2020.

[20] N. Guenther dan M. Schonlau, "Support vector machines," The Stata Journal, vol. 16, no. 4, pp. 917–937, 2016.

[21] E. Osuna, R. Freund, dan F. Girosi, "An improved training algorithm for support vector machines," in Proceedings of IEEE, 1997, pp. 252–1723.

[22] C. Cortes dan V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, pp. 273-297, 1995.

[23] N. Settouti, M. El Habib Daho, M. E. Amine Lazouni, and M. A. Chikh, "Random forest in semi-supervised learning (Co-Forest)," in 2013 8th International Workshop on Systems, Signal Processing and their Applications (WoSSPA), Algiers, Algeria, 2013, pp. 326-329, doi: 10.1109/WoSSPA.2013.6602385.

[24] P. S. Akash, M. E. Kadir, A. A. Ali, M. N. Ahad Tawhid, and M. Shoyaib, "Introducing Confidence as a Weight in Random Forest," in 2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), Dhaka, Bangladesh, 2019, pp. 611-616, doi: 10.1109/ICREST.2019.8644396.

[25] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A Comparative Analysis of XGBoost," Preprint, Nov. 2019.

[26] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), New York, NY, USA, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[27] P. Zhang, Y. Jia, and Y. Shang, "Research and application of XGBoost in imbalanced data," International Journal of Distributed Sensor Networks, vol. 18, no. 6, 2022, doi: 10.1177/15501329221106935.

[28] C. P. Ananda, "Machine Learning Untuk Prediksi Gaya Hidup Berdasarkan Socioeconomic Status Ses Menggunakan Algoritma Catboost Studi Kasus: Mahasiswa UIN Jakarta", Bachelor's thesis, Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta

[29] R. Sanjeetha, A. Raj, K. Saivenu, M. I. Ahmed, B. Sathvik, A. Kanavalli, "Detection and mitigation of botnet based DDoS attacks using catboost machine learning algorithm in SDN environment," International Journal of Advanced Technology and Engineering Exploration, vol. 8, no. 76, p. 445., 2021.

[30] Rizky, P. S., Hirzi, R. H., Hidayaturrohman, U. (2022). Perbandingan Metode LightGBM dan XGBoost dalam Menangani Data dengan Kelas Tidak Seimbang. J Statistika: Jurnal Ilmiah Teori dan Aplikasi Statistika, 15(2), 228-236.

[31] Shehadeh, A., Alshboul, O., Al Mamlook, R. E., Hamedat, O. (2021). Machine learning models for predicting the residual value of heavy construction equipment: An evaluation of modified decision tree, LightGBM, and XGBoost regression. Automation in Construction, 129, 103827.

[32] Nugraha, A. F., Aziza, R. F. A., Pristyanto, Y. (2022). Penerapan metode stacking dan random forest untuk meningkatkan kinerja klasifikasi pada proses deteksi web phishing. Jurnal Infomedia: Teknik Informatika, Multimedia, dan Jaringan, 7(1), 39-44.

[33] Sanjaya, J., Renata, E., Budiman, V. E., Anderson, F., Ayub, M. (2020). Prediksi Kelalaian Pinjaman bank Menggunakan random forest Dan adaptive boosting. Jurnal Teknik Informatika dan Sistem Informasi, 6(1).

[34] Pandey, P., Prabhakar, R. (2016, August). An analysis of machine learning techniques (J48 and AdaBoost)-for classification. In 2016 1st India International Conference on Information Processing (IICIP) (pp. 1-6). IEEE.

[35] Little, R.J.A., Rubin, D.B. (2002). "Statistical Analysis with Missing Data." John Wiley % Sons.

[36] Witten, I.H., Frank, E., & Hall, M.A. (2011). "Data Mining: Practical Machine Learning Tools and Techniques." Morgan Kaufmann.

[37] Han, J., Kamber, M., & Pei, J. (2011). "Data Mining: Concepts and Techniques." Morgan Kaufmann.

[38] Chawla, N.V., Japkowicz, N., & Kotcz, A. (2004). "Editorial: Special Issue on Learning from Imbalanced Data Sets." ACM SIGKDD Explorations Newsletter.

[39] Jolliffe, I.T., & Cadima, J. (2016). "Principal component analysis: a review and recent developments." Philosophical Transactions of the Royal Society A.

[40] I. S. Mangkunegara and P. Purwono, "Analysis of DNA Sequence Classification Using SVM Model with Hyperparameter Tuning Grid Search CV," 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), Malang, Indonesia, 2022.

[41] Anjan, G.S.K.; Kumar Verma, A.; Radhika, S. K-Nearest neighbors and grid search CV based real time fault monitoring system for industries. In Proceedings of the 2019 IEEE 5th International Conference for Convergence in Technology, I2CT 2019, Bombay, India, 29–31 March 2019; p. 5.

[42] T. Yan, S. L. Shen, A. Zhou, and X.-S. Chen, "Prediction of geological characteristics from shield operational parameters using integrating grid search and K-fold cross validation into stacking classification algorithm,", 2022

[43] K. R. Singh, K. P. Neethu, K. Madhurekaa, A. Harita, and P. Mohan, "Parallel SVM model for forest fire prediction,", 2021.

[44] Polipireddy Srinivas, Rahul Katarya, "hyOPTXg: OPTUNA hyperparameter optimization framework for predicting cardiovascular disease using XGBoost," Biomedical Signal Processing and Control, Volume 73, 2022.

[45] Akiba, T., Sano, S., Yanase, T., Ogino, Y., & Ohta, N. (2019, Juni). "Optuna: A distributed hyperparameter optimization framework," In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

[46] D. Marutho, "Perbandingan Metode Naïve Bayes, KNN, Decision Tree pada Laporan Water Level Jakarta," Infokam, vol. II, no. XV, pp. SEPTEMBER, 2019.

[47] S. Lonang, A. Yudhana, and M. K. Biddinika, "Analisis Komparatif Kinerja Algoritma Machine Learning untuk Deteksi Stunting," J. Media Informatika Budidarma, vol. 7, no. 4, pp. 2109-2117, Oct. 2023.

[48] I. J. Sokolova and G. Lapin, "A unified approach to evaluation metrics for binary classification tasks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 12, pp. 2274-2289, Dec. 2009. DOI: 10.1109/TPAMI.2009.113

[49] R. Caruana et al., "Beyond accuracy, precision, and recall: The quest for a comprehensive performance metric in machine learning," in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, Aug. 2014, pp. 239-248. DOI: 10.1145/2670495.2670547

[50] D. Stone, "Cross-validation: A simple and powerful method for evaluation of learning algorithms," Journal of the American Statistical Association, vol. 79, no. 387, pp. 575-583, Sep. 1974.