Kumpulan jurnaL Ilmu Komputer (KLIK)
Volume 09 No.02 2022
ISSN: 2406-7857

HYPER-PARAMETER TUNING PADA *XGBOOST* UNTUK PREDIKSI KEBERLANGSUNGAN HIDUP PASIEN GAGAL JANTUNG

Muhammad Rizky Mubarok[1], Muliadi[2], Rudy Herteno[3]

[1,2,3] Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lambung Mangkurat

Jalan Ahmad Yani Km. 36, Banjarbaru, Kalimantan Selatan, Indonesia

Email : [1]muhammadrizkymubarok@gmail.com, [2]muliadi@ulm.ac.id, [3]rudy.herteno@ulm.ac.id

Abstrak

Prediksi keberlangsungan hidup pasien gagal jantung telah dilakukan pada penelitian untuk mencari tahu tentang kinerja, akurasi, presisi dan performa dari model prediksi ataupun metode yang digunakan dalam penelitian, dengan menggunakan dataset heart failure clinical records. Namun dataset ini memiliki permasalahan yaitu bersifat tidak seimbang yang dapat menurunkan kinerja model prediksi karena cenderung menghasilkan prediksi kelas mayoritas. Pada penelitian ini menggunakan pendekatan level algoritma untuk mengatasi ketidakseimbangan kelas yaitu teknik boosting dengan metode XGBoost lalu digabungkan dengan metode hyper-parameter tuning agar kinerja yang dihasilkan menjadi lebih baik. Selanjutnya model dilatih dengan dataset dan dibandingkan dengan metode lain, hasilnya menunjukkan bahwa XGBoost dengan tree parzen estimator hyper-parameter tuning mengungguli model lain dan mencapai nilai AUC sebesar 0.94, untuk model XGBoost dengan Random Search memperoleh nilai AUC sebesar 0.933, kemudian untuk model XGBoost dengan Grid Search memperoleh nilai AUC sebesar 0.922, lalu untuk model XGBoost tanpa hyper-parameter tuning memperoleh nilai AUC sebesar 0.904.

Keywords: *Ketidakseimbangan Kelas, XGBoost, Konfigurasi Hyper-parameter, Gagal Jantung*

1. Pendahuluan

Keberlangsungan hidup manusia tidak akan pernah terpisah dari organ-organ yang bekerja secara terus-menerus selama manusia itu masih bernafas. Jantung merupakan bagian organ utama dari tubuh manusia dikarenakan tugasnya yang sangat penting sebagai alat pompa dan distribusi darah yang membawa oksigen dan nutrisi yang sangat dibutuhkan oleh manusia. Jantung merupakan salah satu objek vital dan merupakan salah satu garis pertahanan terakhir untuk hidup selain otak [1]

Prediksi keberlangsungan hidup pasien dengan gagal jantung sendiri sudah dilakukan pada penelitian untuk mencari tahu tentang kinerja, akurasi, presisi dan performa dari model prediksi ataupun metode yang digunakan dalam penelitian, dengan dataset yang berasal dari penelitian yang dilakukan oleh Chicco & Jurman [2], akan tetapi permasalahan yang dihadapi pada dataset ini adalah adanya ketidakseimbangan data yang dapat menurunkan kinerja model prediksi karena cenderung menghasilkan prediksi yang tidak sepesifik dan bersifat umum. [3]

Untuk menangani ketidakseimbangan data ada dua pendekatan umum yang digunakan, yang pertama menggunakan pendekatan level data: *resampling*, dan yang kedua menggunakan pendekatan level algoritma: *bagging* dan *boosting*. Menurut [3] untuk menangani masalah dalam ketidakseimbangan kelas (*imbalance class*) salah satunya adalah menggunakan teknik *ensemble (boosting dan bagging)*.

XGBoost atau *Extreme Gradient Boostin*g adalah metode klasifikasi yang basis pembelajarnya adalah pohon keputusan yang menerapkan teknik *ensemble boosting*. Cara kerja metode *boosting* pada umumnya adalah membuat model baru secara bertahap dengan memperhatikan kesalahan model sebelumnya kemudian menambahkan model baru dengan model baru. Untuk meningkatkan kompleksitas model akhir [4].

XGBoost merupakan algoritma yang banyak memiliki parameter dan parameter tersebut memiliki peran yang sangat penting untuk meningkatkan kinerja suatu algoritma, konfigurasi yang tepat pada parameter tersebutdapat meningkatkan kinerja sebuah model, salah satu satu cara untuk melakukan hyper-parameter tuning adalah dengan cara pencarian manual, tetapi pencarian manual akan memakan banyak waktu terlebih apabila sebuah model memiliki parameter yang sangat banyak, pada penelitian ini hyper-parameter tuning yang digunakan adalah *random search, tree parzen estimator* dan *grid search*.

2. Landasan Teori

2.1. XGBoost

Extreme Gradient Boosting atau yang lebih kerap disapa sebagai XGBoost merupakan implementasi lanjutan dari algoritma gradient boosting yang menggunakan decision tree sebagai basis klasifikasi, diperkenalkan pada tahun 2014 dan telah seringkali digunakan karena kecepatan, efisiensi dan skalabilitasnya untuk memecahkan beragam masalah klasifikasi ataupun regresi [4]. Gradient boosting merupakan algoritma yang dapat menemukan solusi optimal untuk berbagai macam masalah. Konsep dasar dari algoritma ini adalah menyesuaikan parameter pembelajaran secara berulang untuk menurunkan cost function. XGBoost menggunakan model yang lebih teratur untuk membangun struktur pohon, sehingga dapat memberikan kinerja yang lebih baik dan mampu mengurangi kompleksitas model untuk menghindari *overfitting* [5].

2.2. Hyperparameter

Kebanyakan algoritma pembelajaran memiliki parameter untuk diatur, dan pengaturan parameter yang berbeda sering menghasilkan model dengan kinerja yang sangat berbeda. Oleh karena itu, evaluasi dan pemilihan model tidak hanya tentang pemilihan algoritma pembelajaran tetapi juga tentang konfigurasi parameter. Proses menemukan *hyper-parameter* yang tepat disebut sebagai *hyper-parameter tuning*

Hyper-parameter adalah parameter yang sudah ditetapkan sebelum proses pembelajaran sebuah model dan bukan parameter yang didapat melalui proses pelatihan. Secara umum, dalam *machine learning*, dibutuhkan optimasi pada *hyper-parameter* dan memilih satu set *hyper-parameter* yang paling optimal untuk meningkatkan kinerja dan efek dari *machine learning* [6].

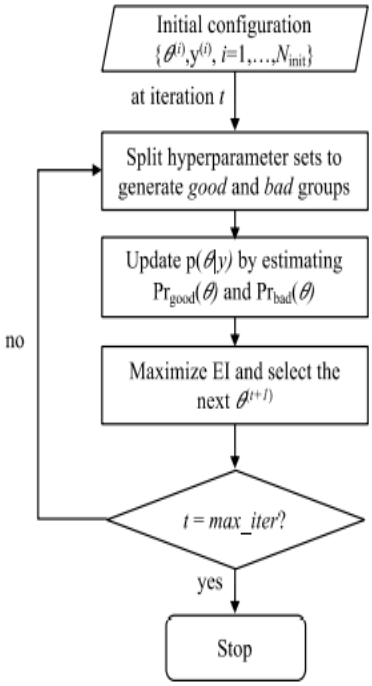
2.3. *Tree Parzen Estimator*

TPE adalah varian dari optimasi Bayesian tradisional. Yang dimana cara kerjanya mengubah ruang konfigurasi menjadi distribusi kerapatan nonparametrik. ruang konfigurasi dapat diwakili oleh distribusi seragam (uniform), distribusi seragam diskrit (quniform) dan distribusi seragam logaritmik (loguniform) [7]. Oleh karena itu, TPE lebih fleksibel daripada optimasi Bayesian tradisional. Pada dasarnya cara kerja TPE mirip dengan cara kerja metode random search karena sama-sama melakukan inisiasi pemilihan pasangan *hyper-parameter* secara acak, namun yang membedakan adalah TPE akan membagi hasil pengamatan *hyper-parameter* menjadi dua grup bagian yaitu, grup baik dan grup buruk berdasarkan persamaan gambar 1 dibawah ini.

$$p(\theta|y) = \begin{cases} \Pr_{good}(\theta) & if\ y < y^* \\ \Pr_{bad}(\theta) & if\ y \geq y^* \end{cases}$$

Gambar 1. Penentuan grup TPE

Penjelasan mengenai gambar 1 adalah, y diasumsikan sebagai nilai pengamatan yang didapat sedangkan y* merupakan nilai pengamatan terdahulu, dalam prosesnya nilai y dan nilai y* akan selalu diperbarui menyeseuaikan berapa banyak iterasi yang ditentukan. Untuk ilustrasi dari cara kerja metode TPE sendiri bisa dilihat pada gambar 6 dibawah ini.



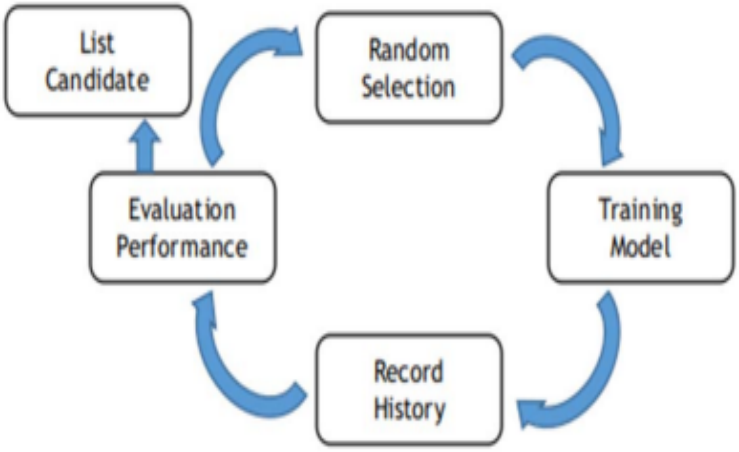
Gambar 2. Alur cara kerja TPE

2.4. *Grid Search*

GridSearch merupakan metode optimasi hyper-parameter yang paling banyak digunakan, karena pada dasarnya cara kerja metode ini menentukan nilai optimal dengan mencari ke semua titik area pencarian yang tersedia, pada kasus regresi Setelah semua kemungkinan nilai pasangan hyper-parameter diuji, pasangan dengan MSE minimum adalah hyper-parameter terbaik. Namun prosesnya sangat membutuhkan banyak waktu dan sangat tidak disarankan jika menggunakan metode ini untuk mencari hyper-parameter pada ruang hyper-parameter yang luas [8].

2.5. *Random Search*

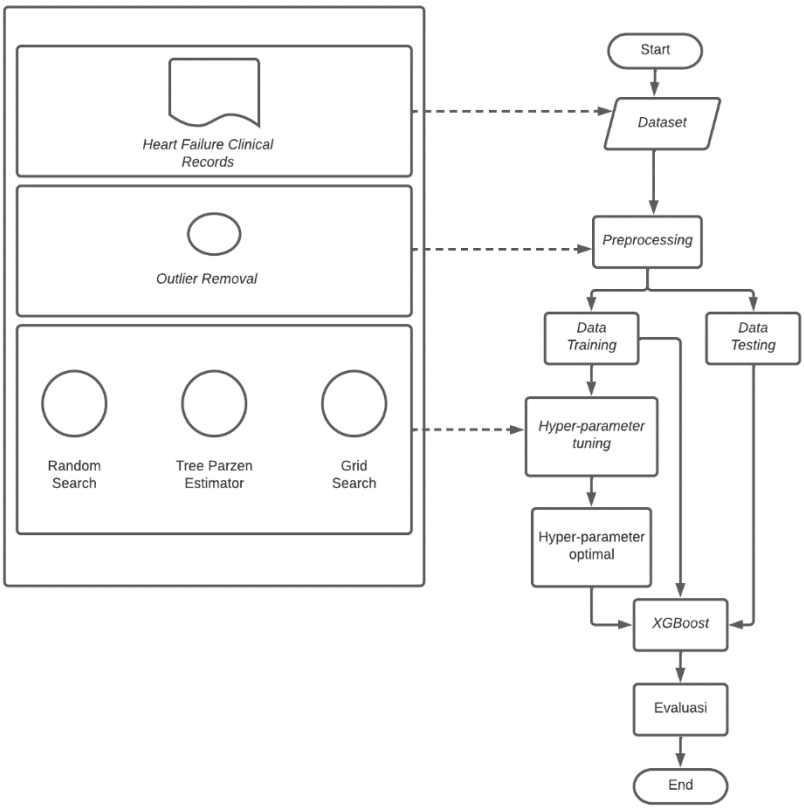
Dibandingkan dengan *Grid Search*, *Random Search* tidak mencoba semua hyper-parameter yang tersedia tetapi dengan melakukan pencarian acak sesuai dengan ruang hyper-parameter yang telah ditetapkan dan berhenti setelah melakukan perulangan sesuai iterasi yang telah diinginkan. Untuk ilustrasi proses pencarian *hyper-parameter* menggunakan metode *random search* bisa dilihat pada gambar 3.



Gambar 3. Ilustrasi metode *random search*

3. Metodologi Penelitian

Alur penelitian dari *Hyper-Parameter Tuning* pada *XGBooost* untuk Prediksi Keberlangsungan Hidup Pasien Gagal Jantung di presentasikan pada **Gambar 1**.



Gambar 4. Alur penelitian.

3.1. Pengumpulan Data

Dataset yang digunakan pada peneltian ini adalah dataset *Heart Failure Clinical Records* yang dirujuk berdasarkan penelitian yang dilakukan oleh Chicco dan jurman [2].

3.2. Preprocessing

Sebelum dilakukan pembagian menjadi *data training* dan *data test*, data akan melalui proses preprocessing terlebih dahulu. *Preprocessing* data yang dilalui pada penelitian ini yaitu, Outlier Removal.

a. *Outlier Removal*

Outlier Removal merupakan proses menghapus beberapa instance yang memiliki karakteristik unik atau memiliki nilai yang ekstrim jika dibandingkan dengan data lainnya sehingga jika data tersebut dipertahankan dapat mempengaruhi hasil dari kinerja model yang akan digunakan, pada penerapannya outlier removal hanya diterapkan pada fitur yang bersifat numerik.

3.3. Modeling

Dataset menjadi dua bagian, yaitu *data training* dan *data testing*. *Data training* digunakan untuk melatih algoritma dalam pembentukan sebuah model, sedangkan

data testing digunakan untuk mengukur kinerja yang didapatkan dari proses pelatihan model dengan *data training*. *Data training* dan *data testing* dibagi dengan proporsi 80% untuk data training dan 20% untuk data testing secara *stratify*, yaitu membagi dataset dengan memperthankan proporsi kelas yang sama.

Tabel 1. Deskripsi Dataset

Data	*Non Deceased*	*Deceased*	Total Instances
Clinical Heart Failure Records	61	163	224

Dalam penelitian ini pemodelan dibagi menjadi empat percobaan, pertama melakukan prediksi menggunakan *XGBoost* tanpa *hyper-parameter tuning*, kedua melakukan prediksi menggunakan *XGBoost* dengan *hyper-parameter tuning* menggunakan *random search*, ketiga melakukan prediksi menggunakan *XGBoost* dengan *hyper-parameter tuning* menggunakan *tree parzen estimator* dan yang keempat melakukan prediksi menggunakan *XGBoost* dengan *hyper-parameter tuning* menggunakan *grid search*. Pada proses pencarian hyper-parameter, metode *hyper-parameter tuning* akan divalidasi menggunakan *5-Fold Cross Validation*, kandidat *hyper-parameter* terbaik akan dipilih berdasarkan nilai tertinggi AUC dari semua kandidat, untuk *hyper-parameter* yang akan dikonfigurasi berdasarkan penelitian terdahulu [9], [10], [11], [12], bisa dilihat pada **Error! Reference source not found.**.

Tabel 2. Hyper-parameter XGBoost

Nama	Deskripsi	Nilai Default
n_estimator	Jumlah pohon individu (tree boosting)	100
max_depth	Kedalaman maksimum dari pohon individu	6
learning_rate	Penyusutan langkah yang digunakan dalam pembaruan model	0.1
min_child_weight	Bobot minimum yang diperlukan untuk membuat sebuah daun pada pohon individu	1
scale_post_weight	Pembagian rasio data pada masing-masing kelas	1

3.4. Evaluasi

Evaluasi merupakan tahap terakhir yang digunakan untuk menilai kinerja sebuah model; pada penelitian ini model akan dievaluasi menggunakan AUC karena menurut Wahono dan Suryana [3] AUC merupakan *metric* yang cocok untuk digunakan pada kasus data tidak seimbang

4. Hasil Dan Pembahasan

4.1. Hasil

4.1.1 *XGBoost* tanpa *Hyper-parameter Tuning*

Klasifikasi dilakukan dengan melibatkan seluruh dataset *Heart failure clinical*

records. Model *XGBoost* yang digunakan menggunakan hyper-parameter default berdasarkan tabel 2. Model dilatih menggunakan data *training* lalu dievaluasi menggunakan data *testing.* Adapun hasil kinerja yang dihasilkan model *XGBoost* tanpa *hyper-parameter tuning* disajikan pada Tabel 3.

Tabel 3. Hasil Evaluasi Model XGBoost

Model	**Nilai AUC**
XGBoost	0.907

4.1.2 *XGBoost* dengan *Random Search Hyper-parameter Tuning*

Konfigurasi *hyperparameter* dengan *random search* dilakukan sebanyak 1000 iterasi sehingga menghasilkan 1000 kandidat hyperparameter secara acak. Untuk *hyper-parameter* yang dikonfigurasi menggunakan *random search* dapat dilihat pada

Tabel 4. Hyper-parameter optimal dengan *random search*

Nama	Domain		Nilai Optimal
	Min	Max	
n_estimator	1	200	53
max_depth	1	20	19
learning_rate	0.01	0.5	0.089
min_child_weight	1	20	7.2
scale_post_weight	1	10	3

Kombinasi hyper-parameter terbaik dengan random search didapat pada iterasi ke 308 dengan rincian *hyper-parameter* yaitu, learning_rate = 0.089, *scale_post_wight* = 3, *min_child_weight* = 7.2, *max_depth* = 19 dan *n_estimator* = 53, konfigurasi *hyper-parameter* akan dilatih ulang menggunakan model *XGBoost* kemudian akan dievaluasi sehingga mendapatkan hasil yang dapat dilihat pada Tabel 5.

Tabel 5. Hasil Evaluasi Model XGBoost dengan random search

Model	**Nilai AUC**
XGBoost dengan random search	0.93

4.1.3 *XGBoost* dengan *Tree Parzen Estimator Hyper-parameter Tuning*

Konfigurasi *hyperparameter* dengan *tree parzen estimator* dilakukan sebanyak 1000 iterasi sama dengan *random search* yang membedakan adalah *tree*

parzen estimator mencari kandidat secara acak lalu membagi kandidat tersebut menjadi dua group bagian, yaitu grup baik dan buruk [7]. Untuk hyper-parameter yang dikonfigurasi menggunakan *tree parzen estimator* dapat dilihat pada Tabel 6.

Tabel 6. Hyper-parameter optimal dengan *tree parzen estimator*

Nama	Domain		Nilai Optimal
	Min	Max	
n_estimator	1	200	74
max_depth	1	20	12
learning_rate	0.01	0.5	0.064
min_child_weight	1	20	7.21
scale_post_weight	1	10	3

Kombinasi *hyper-parameter* terbaik dengan *tree parzen estimator* didapat pada iterasi ke 743 dengan rincian *hyper-parameter* yaitu, *learning_rate = 0.064, scale_post_wight = 3, min_child_weight = 7.2, max_depth = 12* dan *n_estimator = 74,* konfigurasi *hyper-parameter* akan dilatih ulang menggunakan model *XGBoost* kemudian akan dievaluasi sehingga mendapatkan hasil yang dapat dilihat pada Tabel 7.

Tabel 7. Hasil Evaluasi Model XGBoost dengan *tree parzen estimator*

Model	**Nilai AUC**
XGBoost dengan tree parzen estimator	0.944

4.1.4 *XGBoost* dengan *Grid Search Hyper-parameter Tuning*

Konfigurasi *hyperparameter* dengan *grid search* dilakukan sebanyak 4725 iterasi karena cara kerja *grid search* memetakan kombinasi *hyper-parameter* ke dalam sebuah *grid* dan mencoba seluruh kombinasi tanpa ada yang terlewat, semakin luas ruang *hyper-parameter* semakin kompleks proses pencariannya, oleh karena itu *grid search pada* penelitian ini akan dibatasi untuk mengurangi kompleksitasnya berdasarkan penelitian [9]. Untuk *hyper-parameter* yang dikonfigurasi menggunakan *grid search* dapat dilihat pada Tabel 8.

Tabel 8. Hyper-parameter optimal dengan *grid search*

Nama	Domain			Nilai Optimal
	Step	Min	Max	
n_estimator	50	1	200	50

max_depth	2	1	20	2
learning_rate	0.05	0.01	0.5	0.05
min_child_weight	2	1	20	6
scale_post_weight	2	1	10	10

Kombinasi hyper-parameter terbaik dengan grid search didapat pada iterasi ke 11 dengan rincian hyper-parameter yaitu, *learning_rate* = 0.05, *scale_post_weight* = 10, *min_child_weight* = 6, *max_depth* = 2 dan *n_estimator* = 50, konfigurasi *hyper-parameter* akan dilatih ulang menggunakan model XGBoost kemudian akan dievaluasi sehingga mendapatkan hasil yang dapat dilihat pada Tabel 9.

Tabel 9. Hasil Evaluasi Model XGBoost dengan *grid search*

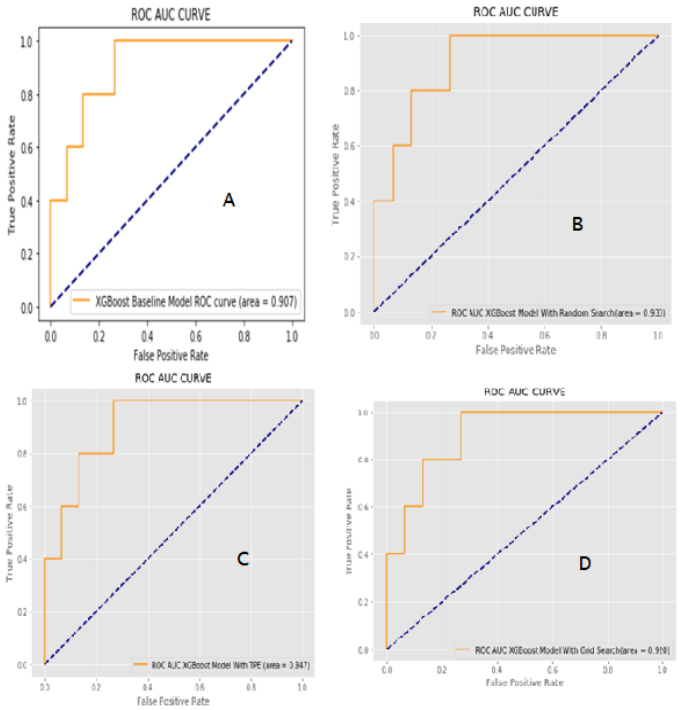
Model	**Nilai AUC**
XGBoost dengan grid search	0.922

4.2. Pembahasan

Efektifvitas dari model menunujukkan peningkatan dengan cara mengoptimalkan *hyper-parameter* yang terdapat pada metode *XGBoost* menggunakan *random search, tree parzen estimator* dan *grid search*, tetapi penelitian ini dilakukan untuk mengetahui kinerja model yang memiliki kinerja terbaik dilihat berdasarkan nilai AUC dalam kasus untuk mengatasi masalah ketidakseimbangan data yang terdapat pada dataset *heart failure clinical records*. Tabel 10 merangkum hasil dari metode yang diusulkan.

Tabel 10. Kinerja Model

Model	AUC
XGBoost tanpa *hyper-parameter tuning*	0.907
XGBoost dengan *random search*	0.933
XGBoost dengan *tree parzen estimator*	0.944
XGBoost dengan *grid search*	0.922



Gambar 5. Kurva ROC-AUC untuk *XGBoost* tanpa hyper-parameter tuning dan *XGBoost* dengan *hyper-parameter tuning*

Gambar 5, adalah visualisasi kurva ROC-AUC yang merepresentasikan performa model yang dihitung berdasarkan klasifikasi data antara true positive rate dan false-positive, AUC untuk *XGBoost* tanpa penyetelan *hyper-parameter* (A) dengan nilai 0,907, *XBoost* dengan *random search* (B) dengan nilai 0.933, *XGBoost* dengan *tree parzen estimator* (C) dengan nilai 0.944 dan *XGBoost* dengan *grid search* (D) dengan nilai 0.922.

5. Kesimpulan

Penelitian ini menyajikan efek hyper-parameter tuning pada *XGBoost* untuk menangani ketidakseimbangan kelas pada dataset *heart failure clinical record*. *Hyper-parameter* terbaik diidentifikasi menggunakan *random search, tree parzen estimator* dan *grid search*. Penelitian ini membuktikan bahwa *XGBoost* dengan *tree parzen estimator* dapat mencapai kinerja yang lebih tinggi berdasarkan nilai AUC terhadap data yang tidak seimbang dibandingkan kombinasi metode dengan *hyper-parameter tuning* yang lain.

Daftar Pustaka

[1] Rozie, F., & Trias Pontia, F. 2016. "Rancang Bangun Alat Monitoring Jumlah Denyut Nadi Jantung Berbasis Android". Jurusan Teknik Elektro Fakultas Teknik Universitas Tanjungpura.

[2] Chicco, D., & Jurman, G. 2020. "Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone". BMC

Medical Informatics and Decision Making, 20(1). https://doi.org/10.1186/s12911-020-1023-5

[3] Wahono, R. S., & Suryana, N. 2013. "Combining particle swarm optimization based feature selection and bagging technique for software defect prediction". International Journal of Software Engineering and Its Applications, 7(5), 153–166. https://doi.org/10.14257/ijseia.2013.7.5.16

[4] Chen, T., & Guestrin, C. 2016. "XGBoost: A scalable tree boosting system". Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, 785–794. https://doi.org/10.1145/2939672.2939785

[5] Tama, B. A., Nkenyereye, L., Islam, S. M. R., & Kwak, K. S. 2020. "An enhanced anomaly detection in web traffic using a stack of classifier ensemble". IEEE Access, 8, 24120–24134. https://doi.org/10.1109/ACCESS.2020.2969428

[6] Dong, H., He, D., & Wang, F. 2020. "SMOTE-XGBoost using Tree Parzen Estimator optimization for copper flotation method classification". *Powder Technology*, *375*, 174–181. https://doi.org/10.1016/j.powtec.2020.07.065

[7] Bergstra, J., Bardenet, R., Bangio, Y., & Kégl, B. 2011. "Algorithms for Hyper-Parameter Optimization". Advances in Neural Information Processing Systems 24 (NIPS 2011), 24.

[8] Huang, Q., Mao, J., & Liu, Y. 2012. "An Improved Grid Search Algorithm of SVR Parameters Optimization". *IEEE*.

[9] Priscilla, C. V., & Prabha, D. P. 2020. "Influence of optimizing xgboost to handle class imbalance in credit card fraud detection". Proceedings of the 3rd International Conference on Smart Systems and Inventive Technology, ICSSIT 2020, 1309–1315. https://doi.org/10.1109/ICSSIT48917.2020.9214206

[10] Wang, Y., & Sherry Ni, X. 2019. "AN XGBOOST RISK MODEL VIA FEATURE SELECTION AND BAYESIAN HYPER-PARAMETER OPTIMIZATION". International Journal of Database Management Systems, 11(01), 01–17. https://doi.org/10.5121/ijdms.2019.11101

[11] Ogunleye, A., & Wang, Q. G. 2020. "XGBoost Model for Chronic Kidney Disease Diagnosis"s. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 17(6), 2131–2140. https://doi.org/10.1109/TCBB.2019.2911071

[12] Gertz, M., Große-Butenuth, K., Junge, W., Maassen-Francke, B., Renner, C., Sparenberg, H., & Krieter, J. 2020. "Using the XGBoost algorithm to classify neck and leg activity sensor data using on-farm health recordings for locomotor-associated diseases". Computers and Electronics in Agriculture, 173. https://doi.org/10.1016/j.compag.2020.105404