ISSN: 2339-2541
JURNAL GAUSSIAN, Volume 11, Nomor 3, Tahun 2022 Halaman 397-406
Online di: https://ejournal3.undip.ac.id/index.php/gaussian/

PENERAPAN *TUNING HYPERPARAMETER RANDOMSEARCHCV* PADA *ADAPTIVE BOOSTING* UNTUK PREDIKSI KELANGSUNGAN HIDUP PASIEN GAGAL JANTUNG

Tita Aulia Edi Putri[1*], Tatik Widiharih[2], Rukun Santoso[3]
[1,2,3] Departemen Statistika, Fakultas Sains dan Matematika, Universitas Diponegoro
***email:** titaauliaauliaa@gmail.com

DOI: 10.14710/j.gauss.11.3.397-406

Article Info:
Received: 2022-07-10
Accepted: 2022-09-15
Available Online: 2023-01-03

Keywords:
Heart Failure; Tuning Hyperparameter; AdaBoost; RandomSearchCV

Abstract: Heart failure is the number one cause of death every year. Heart failure is a pathological condition characterized by abnormalities in heart function, which results in the failure of blood to be pumped to supply metabolic needs of tissues. The application of data mining and computational techniques to medical records can be an effective tool to predict each patient's survival who has heart failure symptoms. Data mining is a process of gathering important information from big data. The collection of important information is carried out through several processes, including statistical methods, mathematics, and artificial intelligence technology. The AdaBoost method is one of the supervised algorithms in data mining that is widely applied to make classification models. Hyperparameter Optimization is selecting the optimal set of hyperparameters for a learning algorithm. AdaBoost has hyperparameters requiring a classification process set, namely learning rate and n_estimators. RandomSearchCV is a random combination method of selected hyperparameters used to train the model. This research uses heart failure patient data collected at the Faisalabad Institute of Cardiology and at the Allied Hospital in Faisalabad (Punjab, Pakistan) from April to December 2015. The research uses learning rate: [-2.2] (log scale), n_estimators start from 10 to 776, and Kfold=5 and produces the best hyperparameters in learning rate=0.01 and n_estimators=443 with an accuracy value of 0.85 and AUC value of 0.897.

1 PENDAHULUAN

Gagal jantung merupakan salah satu penyebab kematian utama setiap tahunnya sebagai penyakit tidak menular (Kemenkes, 2014). Kondisi ini terjadi ketika jantung tidak mampu mensuplai aliran darah sehingga menyebabkan oksigen diangkut ke jaringan dan organ perifer, atau hanya dapat menyuplai oksigen pada tekanan ekspansi yang tinggi (Klabunde, 2015). Dokter mengandalkan klasifikasi fungsional *New York Heart Association* (NYHA) untuk evaluasi kuantitatif perkembangan penyakit jantung. Klasifikasi fungsional *New York Heart Association* (NYHA) terdapat empat kelas mulai dari tidak ada gejala dari aktivitas biasa (kelas I) hingga tahap aktivitas fisik apapun menyebabkan ketidaknyamanan dan gejala terjadi saat beristirahat (Kelas IV). Meskipun digunakan secara luas, tidak ada metode yang konsisten untuk menilai skor NYHA, dan klasifikasi ini gagal memprediksi fitur dasar dengan andal seperti jarak berjalan kaki atau toleransi olahraga pada pengujian formal (Raphael *et al*., 2007).

Jantung memerlukan perlindungan karena hal tersebut mengendalikan salah satu peran penting, yaitu keberlangsungan hidup manusia. Catatan kesehatan elektronik atau disebut juga dengan rekam medis tidak hanya dapat digunakan dalam penelitian, tetapi juga dapat digunakan untuk praktik klinis sebagai sumber informasi untuk mengungkapkan korelasi yang tersembunyi antar data pasien. Penerapan teknik komputasi seperti teknik

machine learning pada rekam medis dapat menjadi alat yang efektif untuk memprediksi kelangsungan hidup setiap pasien yang memiliki gejala gagal jantung.

Machine learning merupakan suatu sistem yang dibangun berdasarkan pembelajaran pada data-data yang besar. Metode *Adaptive Boosting* merupakan salah satu algoritma *supervised* pada *data mining* yang diterapakan secara luas untuk membuat model klasifikasi. Prinsip kerja *AdaBoost* adalah bekerja dengan terlebih dahulu menyusun suatu model *weak learner*, yakni *tree*, dengan memberikan bobot yang sama pada setiap observasi. Jika prediksi salah menggunakan *weak learner* pertama, hal itu memberikan bobot yang lebih tinggi pada observasi yang telah salah prediksi.

AdaBoost memiliki nilai parameter yang perlu diatur untuk mendapatkan model yang optimal, yang disebut *hyperparameter*. Metode *hyperparameter tuning* berhubungan dengan cara kita mengambil sampel calon arsitektur model yang mungkin dari ruang nilai *hyperparameter* yang mungkin. Salah satu metode *hyperparameter* yang dapat digunakan adalah *RandomSearchCV*. *RandomSearchCV* merupakan metode alternatif yang digunakan untuk menemukan parameter terbaik dalam suatu model, sehingga model tersebut dapat memprediksi data secara akurat.

Berdasarkan uraian di atas mengenai betapa pentingnya deteksi dini Gagal Jantung maka penulis melakukan penelitian berupa analisis performa dari *Adaptive Boosting* menggunakan *optimization RandomSearchCV* dalam klasifikasi deteksi awal Gagal Jantung.

2 TINJAUAN PUSTAKA

Data mining adalah metode pemrosesan data untuk menemukan pola tersembunyi dalam data Anda. Hasil pengolahan data dengan metode *data mining* ini dapat digunakan untuk pengambilan keputusan yang akan datang. *Data mining* juga dikenal sebagai *pattern recognition* (Santosa, 2007). *Decision Tree* atau pohon keputusan adalah teknik untuk mengubah fakta besar menjadi pohon keputusan yang mewakili aturan. Prosedur dalam membangun pemodelan klasifikasi *decision tree* adalah sebagai berikut (Brownlee, 2020):

1. Menentukan nilai *entropy*.
   Misalkan S ∈ R menunjukkan variabel acak diskrit yang mengambil nilai s_1, s_2, ..., s_k dengan probabilitas p_1, p_2, ..., p_k masing-masing *entropy* H(S) dari S kemudian didefinisikan oleh ekspresi:

$$Entropy(S) = -\sum_{i=1}^{k} \frac{n_i}{|S|} * \ log_2\left(\frac{n_i}{|S|}\right) \quad (1)$$

   Keterangan:
   S : Variabel target
   n_i : Proporsi sampel pada partisi ke-i dalam S
   k : Banyaknya partisi S

2. Setelah memeroleh nilai *entropy*, selanjutnya ditentukan nilai *information gain*.

$$Gain(S, A) = \ H(S) - \sum_{v \epsilon Values(A)} \frac{S_{A(v)}}{S} \times H\left(S_{A(v)}\right) \quad (2)$$

   Keterangan:
   A : Atribut (variabel yang diuji)
   S : Variabel target
   H(S) : *Entropy* untuk dataset sebelum terjadi perubahan apa pun
   $S_{A(v)}$: Himpunan bagian dari S dengan nilai v untuk atribut A
   $H(S_{A(v)})$: *Entropy* kelompok sampel dengan nilai v untuk atribut A
   v : Setiap nilai yang mungkin untuk atribut A

3. Memilih atribut dengan nilai *information gain* tertinggi dan bentuk simpul yang berisi atribut dengan nilai *information gain* tertinggi G(x_i).
4. Proses penghitungan *information gain* diulangi sampai semua data masuk ke dalam kelas yang sama. Atribut yang dipilih tidak akan lagi dimasukkan dalam perhitungan nilai *information gain*.

Boosting adalah metode pembelajaran *ensemble* yang menggabungkan sekumpulan pembelajar yang lemah menjadi pembelajar yang kuat untuk meminimalkan kesalahan pelatihan. *AdaBoost* adalah algoritma yang ide dasarnya adalah untuk memilih dan menggabungkan satu set pengklasifikasi lemah untuk membentuk klasifikasi yang kuat (Ting dan Zheng, 2009). Penggunaan *AdaBoost* dapat dikombinasikan dengan algoritma klasifikasi lain untuk meningkatkan kinerja klasifikasi. Untuk meningkatkan kinerja, setiap latihan diberi bobot dan bobot tersebut digunakan untuk melatih klasifikasi yang berbeda. Bobot berubah berulang kali berdasarkan kinerja yang diklasifikasikan (Aggarwal, 2015). Prosedur dalam membangun pemodelan klasifikasi *AdaBoost* sebagai berikut (Hastie *et al*., 2008):

1. Menentukan bobot awal w_i setiap amatan pada gugus data latih ($\boldsymbol{x_i}, y_i$) dengan $i = 1,2,3, \dots, N$
$$w_i = \frac{1}{N} \quad (3)$$
2. Pada setiap iterasi m, dengan $m = 1,2,3, \dots, M$ dilakukan hal berikut:
   a. Sampel dataset menggunakan bobot $w_i^{(m)}$ untuk mendapatkan sampel *training* $\boldsymbol{x_i}$.
   b. Menentukan pendugaan klasifikasi menggunakan *weak learner (stump)*, didapat dengan memilih variabel dengan nilai *maximum* dari persamaan (2), pada data sampel *training* $\boldsymbol{x_i}$.
   c. Hitung kesalahan klasifikasi *weak learner* dengan persamaan berikut:
$$err_m = \frac{\sum_{y_i \neq G_m(\boldsymbol{x_i})} w_i^{(m)}}{\sum_{y_i} w_i^{(m)}} \quad (4)$$
   Keterangan:
   y_i : Nilai sebenarnya dari target variabel
   $w_i^{(m)}$: Bobot dari sampel i pada iterasi m
   $G_m(x_i)$: Nilai prediksi yang dihasilkan oleh algoritma *weak learner* (*stump*) saat melatih suatu komponen klasifikasi pada iterasi m
   d. Hitung koefesien α_m dengan persamaan berikut
$$\alpha_m = \frac{1}{2}\ln\left(\frac{1 - err_m}{err_m}\right) \quad (5)$$
   e. *Update* bobot sampel pelatihannya:
$$w_i^{m+1} = w_i^m \exp(-\alpha_m y_i G_m(x_i)), i = 1,2,3, \dots, N \quad (6)$$
   Dengan keterangan:
   y_i : Nilai sebenarnya dari target variabel
   α_m : Tingkat kepercayaan atas prediksi *weak learner* pada iterasi m
   Jika data diklasifikasikan dengan salah, $y_i G_m(x_i) = -1$ dan jika data diklasifikasikan dengan benar, maka $y_i G_m(x_i) = 1$.
3. Dugaan akhir prediksi merupakan total terboboti dugaan prediksi tiap iterasi dengan persamaan:
4. Estimasi akhir prediksi adalah penjumlahan terbobot dari prediksi prediksi untuk setiap iterasi menggunakan persamaan berikut:

$$G(x) = sign\left[\sum_{m=1}^{M} \alpha_m G_m(x_i)\right] \quad (7)$$

Tanda $sign$ ([ ]) memiliki arti bahwa kita mengikutsertakan tanda positif dan negatif ketika melakukan penjumlahan. Penentuan prediksi klasifikasi akan menghasilkan kelas 1 apabila $\sum_{m=1}^{M} \alpha_m \geq 0$ dan kelas -1 untuk selainnya.

Validasi adalah proses mengevaluasi akurasi prediksi suatu model. Validasi adalah penggunaan model yang ada untuk mendapatkan prediksi dan membandingkan hasil yang diperoleh dengan hasil yang diketahui (Gorunescu, 2011). Secara umum, kinerja klasifikasi diukur dengan menggunakan matriks konfusi. *Confusion matrix* memberikan keputusan yang diperoleh dalam pelatihan dan pengujian, dan *confusion matrix* memberikan penilaian kinerja klasifikasi berdasarkan benar atau salahnya objek (Gorunescu, 2011). Menurut Burkov (2020), fungsi dari *confusion matrix* adalah untuk mengukur nilai *accuracy, specificity*, dan nilai *sensitivity* dari model algoritma yang dievaluasi.

a. *Accuracy*

Persentase jumlah prediksi yang benar. Dapat dihitung dengan rumus berikut:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

b. *Specificity*

Spesifisitas mengukur persentase hasil yang benar-benar negatif yang diidentifikasi dengan benar. Spesifisitas dapat dinyatakan dengan persamaan berikut:

$$Specificity = \frac{TN}{TN + FP} \quad (9)$$

c. *Sensitivity*

Sensitivity mengukur proporsi '*true positive*' yang diidentifikasikan dengan benar, yang dihitung dengan menggunakan persamaan:

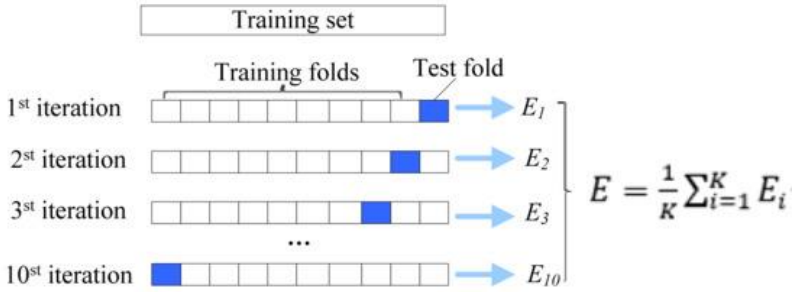
$$Sensitivity = \frac{TP}{TP + FN} \quad (10)$$

Evaluasi hasil prediksi model dianalisis secara visual menggunakan *Receiver Operating Characteristic (ROC) Curve*. Menurut Arian dan Peter (1998), kurva ROC adalah representasi grafis dari hubungan timbal balik antara sensitivitas dan spesifisitas. *Rumus Area Under the ROC Curve (*Altman, *2006;* Fawcett, *2006).*

$$AUC = \frac{1}{2}\sum_{i=1}^{n}(x_{i+1} - x_i)(y_{i+1} - y_i) \quad (11)$$

Dengan x_i mengacu pada nilai *false positive rate (1-Specificity) pada probability cut-offs* yang berbeda dan y_i mengacu pada nilai *true positive rate (Sensitivity)* pada *probability cut-offs* yang berbeda. *false positive rate (1-Specificity)* dirumuskan dalam persamaan (9), *true positive rate (Sensitivity)* dirumuskan dalam persamaan (10).

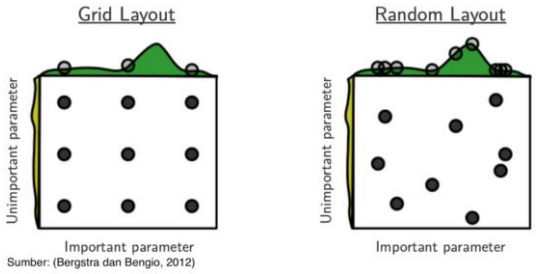
Cross Validation adalah metode estimasi kesalahan prediksi untuk mengevaluasi kinerja model. Salah satu metode validasi silang adalah *k-fold cross validation*. *K-fold cross validation* adalah teknik untuk membagi data menjadi k bagian dengan ukuran yang sama. Penggunaan *k-fold cross validation* untuk menghilangkan bias pada data. Pelatihan dan pengujian dilakukan sebanyak k kali (Tempola et al., 2018). Keakuratan klasifikasi model diperoleh dengan merata-ratakan akurasi setiap iterasi. Menurut Niu *et al*., (2018), alur kerja *cross-validation* diilustrasikan pada Gambar 1.



Gambar 1. Ilustrasi *Cross Validation*

Cross Validation adalah metode estimasi kesalahan prediktif untuk mengevaluasi kinerja model. Salah satu metode validasi silang adalah validasi silang kfold. Validasi K-validasi silang adalah teknik yang membagi data menjadi k bagian yang berukuran sama. Gunakan validasi silang kfold untuk menghilangkan bias data. Pelatihan dan pengujian dijalankan k kali (Tempola et al., 2018). Keakuratan klasifikasi model diperoleh dengan merata-ratakan akurasi setiap iterasi.

Hyperparameter adalah parameter yang nilainya ditetapkan sebelum proses pembelajaran dimulai (Ghawi dan Pfeffer, 2019). Untuk meningkatkan hasil klasifikasi, Anda perlu memilih parameter yang sesuai dalam model yang diusulkan. *Hyperparameter* digunakan untuk mengelola berbagai aspek pembelajaran mesin yang memiliki dampak signifikan pada kinerja dan model yang dihasilkan. Terdapat beberapa jenis *hyperparameter optimization*, diantaranya adalah *GridSearchCV, RandomSearchCV, bayesian optimization, dan evolutionary optimization* (Ghawi dan Pfeffer, 2019). Metode *RandomSearchCV* adalah metode pencarian model terbaik dengan menspesifikasikan nilai *hyperparameter* dari algoritma pembelajaran secara manual. *RandomSearchCV* adalah metode kombinasi acak dari *hyperparameter* dipilih dan digunakan untuk melatih model (Burkov, 2020). Menurut Bergstra dan Bengio (2012), *training flow RandomSearchCV* dapat diilustrasikan pada Gambar 2.



Gambar 2. Ilustrasi Pengaplikasian *Hyperparameter Optimization*

Grid search merupakan pencarian parameter dalam "grid" yang diberikan. Salah satu contoh dari *grid search* yaitu jika jumlah *n_estimators*=[10,20,30,40] dan *learning rate*=[0.01, 0.1, 1.0] maka *grid serach* akan menghitung seluruh kombinasi yang mungkin, yaitu 12 kemungkinan, kemudian dari 12 kemungkinan tersebut dikeluarkan nilai yang memberikan hasil terbaik. *Random Search* berbeda dengan *grid search*, dalam pengerjaannya *random search* hanya mencoba beberapa kemungkinan dari 12 kombinasi kemungkinan yang ada. CV, di akhir kata *RandomSearchCV*, merupakan kepanjangan dari *cross-validation*. Berikut adalah algoritma *Tuning Hyperparameter RandomSearchCV* pada *Adaptive Boosting (*Adaboost*)*.

1. Menentukan *range* parameter *n_estimators* atau banyaknya pohon dan *range learning rate* yang akan diuji.
2. Total *dataset* dibagi menjadi K_{fold} bagian.

3. *Fold* ke-1 adalah ketika bagian ke-1 menjadi data uji (*testing data*) dan sisanya menjadi data latih (*training data*).
4. *Fold* ke-2 adalah ketika bagian ke-2 menjadi data uji (*testing data*) dan sisanya menjadi data latih (*training data*).
5. Demikian sampai *fold* ke-K*fold*.
6. Tahap selanjutnya adalah menghitung akurasi model *Adaptive Boosting (*Adaboost*)* dengan menggunakan persamaan (11) pada setiap *fold* berdasar *range hyperparameter* yang telah ditentukan.
7. Setelah mendapat nilai akurasi (nilai akurasi yang diguanakan adalah AUC) dari semua *fold* dihitung nilai rata-rata aritmatik akurasinya.
8. Nilai *hyperparameter* optimal akan terlihat pada nilai akurasi paling tinggi dari *range hyperparameter* yang ditentukan.

3 METODOLOGI PENELITIAN

Jenis data yang digunakan dalam penelitian ini adalah data sekunder yang disediakan oleh situs resmi *UCI Machine Learning Repository*. Data yang digunakan terdiri dari 299 data pasien gagal jantung yang dikumpulkan di Faisalabad *Heart Disease Institute* dan Faisalabad *Union Hospital* (Punjab, Pakistan) pada bulan April 2015. Variabel dalam penelitian ini terdiri dari satu variabel respon (Y) dan dua belas variabel penjelas (X). Pengolahan data dilakukan menggunakan *Google Colaboratory* (*Colab*). *Colab* adalah pengembangan *environment Python* yang dijalankan dalam *browser* menggunakan *Google Cloud*. Data yang telah diperoleh kemudian dianalisis sebagai berikut:

1. Memasukkan data ke dalam *Colab* dari *Google Drive*.
2. Melakukan pembagian data latih dan data uji dengan proporsi 80:20.
3. Menentukan *range* parameter banyaknya pohon (*n_estimators*) dan *learning rate*.
4. Membuat model *Adaptive Boosting* (*AdaBoost*)
5. *Tuning Hyperparameter Adaptive Boosting (AdaBoost).*
   a. Melakukan pelatihan pada data latih dengan *cross validation* sehingga menemukan parameter banyaknya pohon dan *learning rate* yang optimal.
   b. Membuat model *Adaptive Boosting* (*AdaBoost*) yang telah di-*tuning*.
6. Melakukan evaluasi model *Adaptive Boosting (AdaBoost)* yang telah di-*tuning* menggunakan data uji dengan pengukuran akurasi, sensitivitas, spesifisitas, dan AUC.
7. Interpretasi hasil.

4 Hasil dan Pembahasan

Data dibagi menjadi dua bagian, yaitu data *training* dengan proporsi 80% dan data *testing* dengan proporsi 20%. 239 data latih akan digunakan untuk pemodelan dan 60 data uji akan digunakan untuk menghitung pengukuran akurasi klasifikasi. Pembagian data dilakukan secara *random* dengan stratifikasi pada variabel terikat (Y). Pembagian *dataset* dilakukan menggunakan *software Python* dengan *syntax* sebagai berikut:

```
y= dataset['DEATH_EVENT'].values
X=dataset.drop(['DEATH_EVENT'],axis=1).values
seed=4
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=seed,stratify=y)
```

Dalam penelitian ini, peneliti membatasi *hyperparameter* yang akan diuji yaitu untuk *learning rate*: [-2,2] (*log-scale)* (*learning rate* dipilih berdasarkan jurnal yang ditulis oleh Rijn dan Hutter (2018) yang memberikan hasil paling optimal) dan *n estimators* dimulai dengan 10 sampai 776 (*n estimators* dipilih berdasarkan jurnal yang ditulis oleh Gao dan Liu

(2018) yang memberikan hasil paling optimal). Pembatasan *range hyperparameter* yang akan diuji dapat dilakukan dengan *syntax* sebagai berikut:

```
ada_p_dist={'learning_rate':[10**i for i in range(-2,2)],
        'n_estimators':range(10,776)}
```

Tahap selanjutnya ialah melakukan inisialisasi model *AdaBoost* yang diikuti dengan *train* data *training* menggunakan *RandomSearchCV* dan *cross validation*. *K-fold cross validation* yang digunakan adalah K_{fold}= 5 dikarenakan oleh minimnya data sampel. Hal tersebut dilakukan agar setiap model mempunyai jumlah data yang cukup untuk *validation set*. Inisialisasi model *AdaBoost* dan seleksi *hyperparameter* menggunakan metode *RandomSearchCV* dan *cross validation* dapat dilakukan dengan *syntax* sebagai berikut:

```
model = AdaBoostClassifier()
random_search=RandomizedSearchCV(model,param_distributions=ada_p_dist,scoring='roc_auc',n_jobs=-
1,cv=5,verbose=3,random_state=2)
random_search.fit(train_X,train_y)
```

Tahap verifikasi *hyperparameter* terbaik dilakukan dengan *syntax* sebagai berikut:

```
random_search.best_params_

{'learning_rate': 0.01, 'n_estimators': 443}
```

Berdasarkan *Score CV* terbaik, didapatkan *hyperparameter* terbaik adalah *learning rate*=0,01 dan *n estimators*=443 dikarenakan *learning rate*=0,01 dan *n estimators*=443 mendapatkan *score* atau rata-rata akurasi *roc_auc* tertinggi sebesar 0,897. Pembuatan model *AdaBoost* yang telah di-*tuning* dan *train* data *training* menggunakan model terbaik dilakukan dengan *syntax* sebagai berikut:

```
model_terbaik = AdaBoostClassifier(learning_rate=0.01, n_estimators=443)
model_terbaik.fit(train_X,train_y)
```

Tahap prediksi pada data *testing* menggunakan model terbaik menggunakan *syntax* sebagai berikut:

```
predictions_model_terbaik = model_terbaik.predict(test_X)
```

Tahap evaluasi model yang telah di-*tuning* menggunakan *syntax* sebagai berikut:

```
cm = confusion_matrix(test_y, predictions_model_terbaik)
sns.heatmap(cm, annot = True)
```

Matriks konfusi metode *AdaBoost* dengan *Tuning Hyperparameter RandomSearchCV* dan menggunakan *range hyperparameter learning rate*: [-2,2] (*log-scale)* serta *n estimators* dimulai dari 10 sampai 776, *cross validation* K_{fold}=5 *scoring=roc_auc*, yang dapat dilihat pada Tabel 1.

Tabel 1. Matriks Konfusi *AdaBoost* dengan *Tuning RandomSearchCV*

TRUE	*Predicted*		Total
	Negatif	Positif	
Negatif	38	3	41
Positif	6	13	19
Total	44	16	60

Ukuran-ukuran ketepatan klasifikasi dapat dihitung berdasarkan Tabel 8 dengan perhitungan sebagai berikut:

TP = 13
TN = 38
FN = 6
FP = 3

Akurasi $= \frac{TP+TN}{(TP+TN+FP+FN)} \times 100\% = \frac{13+38}{(13+38+3+6)} \times 100\% = 85\%$

Sensitivitas $= \frac{TP}{(TP+FN)} \times 100\% = \frac{13}{(13+6)} \times 100\% = 68{,}4\%$

Spesifisitas $= \frac{TN}{(TN+FP)} \times 100\% = \frac{38}{(38+3)} \times 100\% = 92{,}6\%$

Rata-rata akurasi untuk metode *AdaBoost* dengan *Tuning Hyperparameter RandomSearchCV* dan menggunakan batasan *hyperparameter learning rate*: [-2,2] (*log-scale)* serta *n estimators* dimulai dari 10 sampai 776, *cross validation* K_{fold}=5 *scoring=roc_auc* adalah sebesar 85%. Nilai sensitivitas sebesar 68,4% menunjukan bahwa tedapat 68,4% pasien dengan kelas asli negatif (selamat) yang diprediksikan secara benar untuk didiagnosis negatif (selamat). Nilai spesifisitas sebesar 92,6% berarti terdapat 92,6% pasien dengan kelas asli positif (meninggal) yang diprediksikan secara benar untuk didiagnosis positif (meninggal).

Ukuran ketepatan klasifikasi atau performa model juga dihitung menggunakan nilai AUC. Nilai AUC dari ROC *Curve* didapatkan dengan *syntax* sebagai berikut:

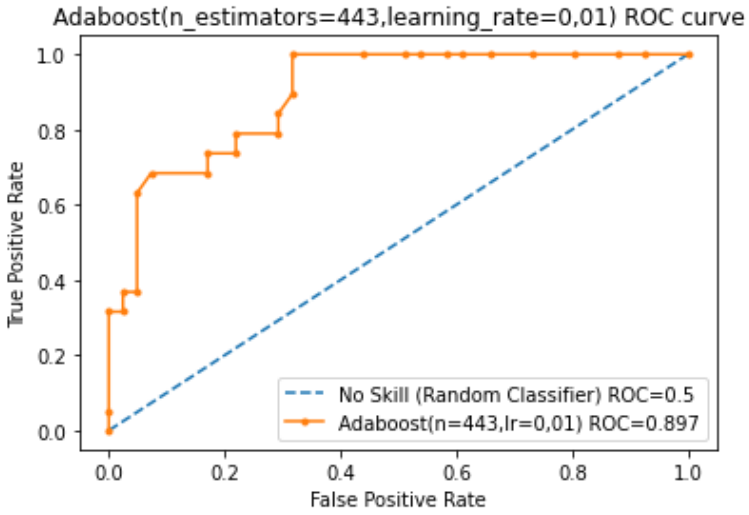
```
modelterbaik_peluangg = random_search.predict_proba(test_X)
modelterbaik_peluangg = modelterbaik_peluangg[:, 1]
roc_auc = roc_auc_score(test_y, pos_peluangg)
print('XGBoost ROC AUC %.3f' % roc_auc)

XGBoost ROC AUC 0.897
```

Berdasarkan *syntax* di atas, maka didapatkan nilai AUC sebesar 0,897 sehingga dapat disimpulkan bahwa kriteria model klasifikasi tersebut adalah *good classification* (nilai AUC berada pada *range* 0,80-0,90 seperti yang disajikan pada tabel 2). *ROC Curve* merupakan kurva untuk memvisualisasikan *tradeoff* antara *true positive rate* (sensitivitas) dan *true negative rate* (1-spesifisitas). *ROC Curve* didapat dari *syntax* sebagai berikut:

```
ns_peluangg = [0 for _ in range(len(test_y))]
modelterbaik_peluangg = random_search.predict_proba(test_X)
modelterbaik_peluangg = modelterbaik_peluangg[:, 1]
ns_auc = roc_auc_score(test_y, ns_peluangg)
AdaBoost_auc = roc_auc_score(test_y, modelterbaik_peluangg)
```

```
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('AdaBoost: ROC AUC=%.3f' % (AdaBoost_auc))
ns_fpr, ns_tpr, _ = roc_curve(test_y, ns_peluangg)
AdaBoost_fpr, AdaBoost_tpr, _ = roc_curve(test_y, modelterbaik_peluangg)
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label=f'No Skill (Random Classifier) ROC={ns_auc}')
pyplot.plot(AdaBoost_fpr, AdaBoost_tpr, marker='.', label=f'AdaBoost(n=443,AdaBoost=0,01)
ROC={round(AdaBoost_auc,3)}')
pyplot.title('AdaBoost(n_estimators=443,learning_rate=0,01) ROC curve')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.show()
```

Berdasarkan *syntax* di atas, maka didapatkan *ROC Curve AdaBoost RandomSearchCV (n_estimators=443,learning rate=0,01)* pada Gambar 1.



Gambar 3. *ROC Curve AdaBoost RandomSearchCV*

Garis jingga pada kurva ROC menunjukkan *trade-off* antara sensitivitas (atau TPR) dan spesifisitas (1 – FPR) pada klasifikasi data *testing*. Garis biru pada kurva ROC mewakili pengklasifikasi dengan tingkat kinerja acak. Sebagai dasar, pengklasifikasi acak diharapkan memberikan titik-titik yang terletak di sepanjang diagonal (FPR = TPR). Semakin dekat kurva ke diagonal 45 derajat dari ruang ROC, semakin kurang akurat pengujiannya. Pengklasifikasi dengan tingkat kinerja sempurna menunjukkan kombinasi dua garis lurus dari titik asal (0.0, 0.0) ke sudut kiri atas (0.0, 1.0) dan selanjutnya ke sudut kanan atas (1.0, 1.0).

5 KESIMPULAN

Berdasarkan hasil dan pembahasan pada bab sebelumnya, diperoleh kesimpulan sebagai berikut:

1. Hasil analisis menunjukkan bahwa kombinasi *hyperparameter* terbaik yang diperoleh dalam penelitian ini dari proses *tuning* menggunakan *RandomSearchCV* adalah *n_estimators* sebesar 443 dan *learning rate* sebesar 0,01.
2. Metode *AdaBoost* yang diterapkan menghasilkan akurasi klasifikasi 85%, nilai spesifisitas sebesar 92,6%, nilai sensitivitas sebesar 68,4% dan nilai AUC sebesar 89,7% pada data uji. Nilai AUC sebesar 89,7% ≈ 0,897 berada di antara 0,80–0,90 yang menjadikan kriteria model klasifikasi tersebut menjadi *good classification*. Berdasarkan hal tersebut, *RandomSearchCV* dapat digunakan untuk *Tuning Hyperparameter* pada

metode *AdaBoost*. Pencapaian ini relatif sangat baik sehingga dapat disimpulkan bahwa model dapat mengklasifikasi dan mendiagnosis status keselamatan pasien dengan baik.

DAFTAR PUSTAKA

Aggarwal, C. C. 2015. *Data Mining*. New York: Springer.

Altman, D.G. 2006. *Practical Statistics for Medical Research*. Chapman dan Hall/CRC.

Arian R. V. E., dan Peter M. Th. P. 1998. Receiver operating characteristic (ROC) analysis: Basic principles and applications in radiology. European Journal of Radiology, pp. 88-94. https://doi.org/10.1016/S0720-048X(97)00157-5.

Bergstra, J., dan Bengio, Y. (2012). *Random search for hyper-parameter optimization.* Journal of Machine Learning Research, 13, 281–305. https://www.researchgate.net/publication/262395872_Random_Search_for_Hyper-Parameter_Optimization.

Brownlee, J. (2020). *Probability for Machine Learning : Discover how to harness uncertainty with Python* (v1.9).

Burkov, A. (2020). *The Hundred-Page Machine Learning Book. In Journal of Information Technology Case and Application Research* (Vol. 22, Issue 2). https://doi.org/10.1080/15228053.2020.1766224.

Fawcett, T. 2006. *An Introduction to ROC Analysis*. Pattern Recogn. Lett., Volume 27(8), pp. 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Ghawi, R., dan Pfeffer, J. (2019). *Efficient Hyperparameter Tuning with Grid Search for Text Categorization using kNN Approach with BM25 Similarity*. Open Computer Science, 9(1), 160–180. https://doi.org/10.1515/comp-2019-0011.

Gorunescu, F. (2011). *Data Mining: Concepts, Models and Techniques*. Berlin: Springer.

Hastie T., Tibshirani R., dan Friedman J. 2008. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Ed ke-2. New York (NY): Springer Verlag.

Kasron. 2012. *Kelainan dan Penyakit Jantung Pencegahan serta Pengobatannya.* yogyakarta: Nuha Medika.

Kemenkes RI. (2014). *Situasi kesehatan jantung*. In Pusat data dan informasi kementerian kesehatan RI. https://doi.org/10.1017/CBO9781107415324.004

Klabunde, R. E. (2015). *Konsep Fisiologi Kardiovaskular*. Ed. 2. Jakarta: EGC.

Niu, M., Li, Y., Wang, C., & Han, K. (2018). RFAmyloid: A web server for predicting amyloid proteins. *International Journal of Molecular Sciences*, *19*(7). https://doi.org/10.3390/ijms19072071

Raphael, C., Briscoe, C., Justin, D. Z. I. W., Manisty, C., Sutton, R., Mayet, J., Francis, D. P. (2007). *Limitations of the New York Heart Association functional classification system and self-reported walking distances in chronic heart failure*. Heart ; 93(4):476–82.

Santosa, B. 2007. *Data Mining: Teknik Pemanfaatan Data untuk Keperluan Bisnis*. Graha Ilmu. Yogyakarta.

Tempola, F., Muhammad, M., dan Khairan, A. (2018). *Perbandingan Klasifikasi Antara KNN dan Naive Bayes pada Penentuan Status Gunung Berapi dengan K-Fold Cross Validation.* Jurnal Teknologi Informasi Dan Ilmu Komputer, 5(5), 577. https://doi.org/10.25126/jtiik.201855983.

Ting, K., Zheng, Z. (2009). *A Study of AdaBoost with Naive Bayesian Classifiers: Weakness and Improvement.* Computational Intellegence.19(2),186-200.