e-ISSN : 2830-6252
p-ISSN : 2830-6562

JCI, Vol. 2 No. 2 (2022) 15 – 25 | *https://doi.org/10.54066/jci.v2i2.238*

luceant opera vestra bona

Jurnal Cakrawala Informasi

Journal Homepage: http://www.itbsemarang.ac.id/sijies/index.php/jci
e-Mail: jci@itbsemarang.ac.id

Analisa Perbandingan 7 Algoritma Klasifikasi Menggunakan *Dataset* Sensus Penduduk

Tri Santosa Wijanarko

Program Studi Bisnis Digital, Sekolah Tinggi Ilmu Ekonomi (STIE) Widya Manggala Semarang

INFO ARTIKEL

Histori artikel:

Diterima : 10 Desember 2022
Revisi : 13 Desember 2022
Disetujui : 25 Desember 2022
Publikasi : 30 Desember 2022

Kata kunci:
Algoritma Klasifikasi
Rapidminer
Dataset Adult
Decision Stump
Decision Tree
Naive Bayes
k-*Nearest Neighbor*
Random Forest
Rule Based

ABSTRACT

In this study the authors tried to make a comparison of the seven classification algorithms by taking population datasets from published data. The purpose of this research is to compare the performance of classification techniques using Rapidminer software. In testing using an adult dataset which has 35,500 instances including 15 attributes (6 continuous 6 nominal). The adult dataset contains data about adults such as age, gender, ethnicity, status, education, and so on. The data in the adult dataset are classified based on salaries that exceed Rp. 50,000 and a salary equal to or less than Rp. 50,000. Researchers try to make a comparison of the seven classification algorithms to get the accuracy of each of these algorithms. The algorithms used are Decision Tree (C4.5), Decision Stump, Random Tree, Random Forest, Naive Bayes, k-Nearest Neighbor, and Rule Based. For the validation method the author uses k-Fold Cross Validation while to find out the accuracy of the algorithm between one another the author uses a parametric different test using a T-Test.

ABSTRAK

Dalam penelitian ini penulis mencoba melakukan komparasi dari tujuh algoritma klasifikasi dengan mengambil *dataset* kependudukan dari data yang sudah dipublikasikan. Tujuan dari penelitian ini adalah mengetahui perbandingan performa teknik klasifikasi menggunakan *software Rapidminer*. Dalam pengujian menggunakan *dataset adult* yang memiliki 35.500 *instance* meliputi 15 attribut (6 *continous* 6 nominal). *Dataset adult* berisi data mengenai orang dewasa seperti umur, jenis kelamin, etnis, status, edukasi, dan lain-lain. Data-data pada *dataset adult* diklasifikasikan berdasar gaji yang melebihi Rp. 50.000 dan gaji yang sama dengan atau kurang dari Rp. 50.000. Peneliti mencoba melakukan komparasi dari tujuh algoritma

* Korespondensi penulis: tri.santosa.wijanarko@gmail.com

klasifikasi untuk mendapatkan akurasi dari masing-masing algoritma tersebut. Algoritma yang digunakan adalah *Decision Tree* (C4.5), *Decision Stump, Random Tree, Random Forest, Naive Bayes,* k-*Nearest Neighbor,* dan *Rule Based*. Untuk metode validasi penulis menggunakan k-*Fold Cross Validation* sedang untuk mengetahui akurasi algoritma antara satu dengan yang lain penulis menggunakan uji beda parametik dengan menggunakan t-*test*.

PENDAHULUAN

Ada beberapa masalah yang sering ditemukan dalam menganalisa data sensus kependudukan diantaranya data tersebut mempunyai jumlah dan dimensi data yang cukup besar. Untuk mendapatkan analisa akhir semua prosedur diperlukan, akan tetapi pengujian yang terlalu banyak dapat menyulitkan untuk mendapatkan hasil akhir. Dalam masalah ini *machine learning* dapat membantu untuk mengatasi masalah tersebut. Penelitian untuk komparasi algoritma klasifikasi dengan skala besar menggunakan *machine* learning Weka pernah dilakukan pada pengujian menggunakan parameter *Kappa Stastistic*, *Mean AbsolutE Error*, dan *Root Mean Squared Error* [1]. Perbandingan pengujian atau komparasi tujuh algoritma klasifikasi terhadap *dataset adult*, peneliti menggunakan algoritma *Decision Tree* (C4.5), *Decision Stump*, *Random Tree*, *Random Forest*, *Naive Bayes*, k-*Nearest Neighbor*, *Rule Based* dengan metode validasi k-*Fold Validation* dan uji beda parametik menggunakan t-*test*.

TINJAUAN PUSTAKA

Penelitian untuk komparasi algoritma klasifikasi dengan skala besar menggunakan *machine learning* Weka pernah dilakukan pada pengujian menggunakan parameter *Kappa Statistic*, *Mean Absolut Error*, dan *Root Mean Squared Error* [1].

Peneliti yang melakukan komparasi algoritma klasifikasi lainnya adalah Romi Satria Wahono untuk prediksi cacat *software* dengan melakukan komparasi sepuluh algoritma klasifikasi antara lain LR, LDA, NB, k-NN, NN, SVM, *Decision Tree* (C4.5), *Chart*, dan RF, menggunakan metode validasi *Stratified* 10-*Fold Cross Validation* dan metode perbandingan non parametik *Friedman* dan *Nemenyi Post Hoc Test* untuk mengetahui perbedaan signifikan dari nilai *Area Under Curve* dari sepuluh algoritma tersebut [2].

Data Mining

Informasi tentang sebuah objek, peristiwa atau kejadian, konsep yang didokumentasikan sering disebut dengan data. Sebuah data atau sekumpulan data tidak akan mempunyai arti apa-apa bilamana data hanya dikumpulkan begitu saja. Ibarat di padang rumput yang luas kita tidak pernah tahu apa yang ada di dalamnya. Padahal di dalam padang rumput itu ada sesuatu yang berharga misalnya ada singa di dalamnya. Proses mencari singa di dalam hamparan padang rumput yang luas atau proses pencarian data yang mempunyai informasi yang sangat penting disebut juga *data mining*. *Data mining* adalah proses untuk menemukan pola (*pattern*) dari suatu data. Pola (*pattern*) yang ditemukan harus memiliki arti atau mengandung informasi penting [3][4].

Algoritma C4.5

Algoritma C4.5 termasuk dalam kelompok algoritma *Decision Tree*. Diperkenalkan oleh J. Ross Quinlan diakhir tahun 1970 hingga awal tahun 1980-an. J. Ross Quinlan seorang peneliti di bidang mesin pembelajaran yang merupakan pengembangan dari algoritma ID3 (*Interative Dichotomiser*). Dalam beberapa penelitian algoritma C4.5 ini menjadi pilihan terbaik dibandingkan dengan beberapa algoritma klasifikasi lain.

Algoritma *Decision Tree* (C4.5) merupakan algoritma klasifikasi yang mengklasifikasikan sampel data secara *topdown* mulai dari simpul akar dan bergerak sesuai dengan hasil pengujian dari node internal sampai node cabang dicapai dan *class* ditetapkan [5]. Dalam *Decission Tree* ini data yang berupa fakta dirubah menjadi sebuah pohon keputusan yang berisi aturan dan tentunya dapat lebih mudah dipahami dengan bahasa alami. Model pohon keputusan banyak digunakan pada kasus data dengan *output* yang bernilai diskrit. Walaupun tidak menutup kemungkinan dapat juga digunakan untuk kasus data dengan atribut numerik. Setiap *node* dalam *Decision Tree* merepresentasikan sebuah atribut. Sedangkan cabang dari *node* merupakan nilai dari atribut tersebut, serta daun merepresentasikan kelas. *Node* paling atas pada *Decision Tree* disebut sebagai *root node*. *Root node* ini tidak memiliki *input* serta bisa saja tidak memiliki *output* dan bahkan dapat memiliki *output* lebih dari satu. Internal *root* merupakan *node* percabangan yang hanya memiliki satu *input* dan memiliki minimal dua *output*. *Leaf node* atau terminal *node* merupakan *node* akhir yang hanya memiliki satu *input* serta tidak memiliki *output*. *Root node* atau atribut akar disimbolkan dengan persegi tumpul yang berada paling puncak yaitu *outlook*. Cabang disimbolkan dengan garis dan *leaf node* atau *terminal node* disimbolkan dengan persegi berujung yang berisi label atau tujuan. Langkah untuk membuat sebuah *Decision Tree* dari algoritma C4.5 adalah sebagai berikut:

1. Mempersiapkan *data training*. *Data training* yaitu data yang diambil dari data histori yang pernah terjadi sebelumnya atau disebut data masa lalu dan sudah dikelompokan dalam kelas-kelas tertentu.
2. Menentukan akar pohon. Akar pohon ditentukan dengan cara menghitung *Gain Ratio* tertinggi dari masing-masing atribut. Sebelum menghitung *Gain Ratio*, terlebih dahulu menghitung *Total Entropy* sebelum dicari masing-masing *Entropy Class*, adapun rumus mencari *Entropy* sebagai berikut:

$$Entropy(S) = -\sum p \log 2\, p$$

S = Himpunan *dataset* kasus

P = Probabilitas yang diperoleh dari jumlah kasus pada partisi dibagi total kasus

Rumus untuk menghitung gain adalah sebagai berikut:

$$\text{Gain (S, A)} = Entropy(S) - \sum_{i-1}^{n} \frac{|S_i|}{|S|} * Entropy(S_i)$$

S = Himpunan kasus

A = Atribut

n = Jumlah atribut A

|Si| = Jumlah kasus pada partisi ke-i

|S| = Jumlah kasus dalam S

Atribut dengan *Information Gain* tertinggi dipilih sebagai atribut node awal (*root node*) serta cabang-cabangnya dibuat sesuai nilai-nilai kemungkinan, proses tersebut akan terus berulang pada setiap cabang.

Algoritma *Decision Stump*

Decision Stump adalah model *machine learning* yang terdiri dari pohon keputusan satu tingkat yang mempunyai arti pohon keputusan dengan satu simpul internal (akar) dimana langsung tersambung dengan *node* terminal (daunnya). Algoritma *Decision Stump* membuat prediksi berdasarkan nilai hanya pada fitur *input* tunggal. Tergantung pada jenis fitur *input* yang mempunyai pilihan beberapa variasi. Untuk fitur nominal, memungkinkan untuk membangun satu pohon yang berisi daun atau cabang untuk setiap nilai fitur atau satu pohon dengan dua daun atau dua cabang, satu yang sesuai dengan beberapa kategori yang dipilih, dan cabang lain untuk semua kategori lainnya. Pohon yang dihasilkan dapat digunakan untuk mengklasifikasi contoh yang tak terlihat. Operator ini bisa sangat efisien ketika didukung dengan operator seperti operator *AdaBoost*. *Dataset* memiliki beberapa atribut dan sebuah kelas. *Node* cabang/daun dari pohon keputusan berisi nama kelas sedangkan *node* non-cabang/daun adalah simpul keputusan. *Node* keputusan adalah tes atribut dengan masing-masing cabang (pohon keputusan lain) menjadi sebuah nilai dari atribut.

Algoritma *Random Tree*

Algoritma ini mirip dengan *Decision Tree* hanya saja untuk setiap pemecah hanya atribut *random subset* yang tersedia. Operator ini memakai data nominal dan numerik. Sedangkan *Decision Tree* merupakan metode klasifikasi yang dapat dengan mudah dipahami. Operator *Random Tree* bekerjasama dengan *Quinlans* C4.5 atau CART yang menggunakan atribut *random subset* sebelum diterapkan. Ukuran *subset* ditentukan oleh bagian parameter *ratio* [6].

Representasi data sebagai pohon memiliki keunggulan dibandingkan dengan pendekatan lain menjadi bermakna dan mudah untuk menafsirkan. Tujuannya adalah untuk menciptakan sebuah model klasifikasi yang memprediksi nilai label berdasarkan beberapa masukan atribut *ExampleSet*. Setiap *node* interior pohon sesuai dengan salah satu atribut masukan. Jumlah tepi *node* interior adalah sama dengan jumlah nilai yang mungkin dari atribut *input* yang sesuai. Setiap *node* daun mewakili nilai dari label yang diberikan nilai-nilai atribut masukan diwakili oleh jalan dari akar ke daun.

Deskripsi ini dapat dengan mudah dipahami dengan mempelajari contoh proses operator pohon keputusan. Pemangkasan adalah teknik di mana *node* daun yang tidak menambah kekuatan diskriminatif dari pohon keputusan dikeluarkan. Hal ini dilakukan untuk mengubah sebuah pohon lebih spesifik atau lebih pas untuk bentuk yang lebih umum dalam rangka meningkatkan daya prediksi pada *dataset* yang tak terlihat. Pra pemangkasan adalah jenis pemangkasan paralel dilakukan untuk proses pembuatan pohon. Pasca pemangkasan, di sisi lain, dilakukan setelah proses pembuatan pohon selesai.

Algoritma *Random Forest*

Random Forest adalah contoh dari model *ensemble*, yaitu model yang dibentuk oleh satu set model sederhana [7]. Secara khusus, *Random Forest* terdiri dari satu set pohon keputusan, baik klasifikasi atau regresi pohon, tergantung pada masalah yang ditangani. Pengguna memutuskan jumlah pohon di *ensemble*. Setiap pohon yang dipelajari menggunakan sampel *bootstrap* yang diperoleh secara acak dengan menggambar kasus N dengan penggantian dari *dataset* asli, dimana N adalah jumlah kasus dalam *dataset* itu. Dengan masing-masing *data training*, pohon yang berbeda diperoleh. Setiap *node* dari pohon-pohon ini dipilih berdasarkan prediktor yang diacak. Ukuran *subset* ini harus jauh lebih kecil dari jumlah prediktor dalam *dataset* [8].

Metode *Random Forest* adalah pengembangan dari metode CART, yaitu dengan menerapkan metode *bootstrap aggregating* (*bagging*) dan *random feature selection*. Dalam *Random Forest*, banyak pohon ditumbuhkan sehingga terbentuk hutan (*forest*), kemudian analisis dilakukan pada kumpulan pohon tersebut. Pada gugus data yang terdiri atas n amatan dan p peubah penjelas, *Random Forest* dilakukan dengan cara:

1. Lakukan penarikan contoh acak berukuran n dengan pemulihan pada gugus data. Tahapan ini merupakan tahapan *bootstrap*.
2. Dengan menggunakan contoh *bootstrap*, pohon dibangun sampai mencapai ukuran maksimum (tanpa pemangkasan). Pada setiap simpul, pemilihan pemilah dilakukan dengan memilih m peubah penjelas secara acak, dimana m < p. Pemilah terbaik dipilih dari m peubah penjelas tersebut. Tahapan ini adalah tahapan *random feature selection*.
3. Ulangi langkah 1 dan 2 sebanyak k kali, sehingga terbentuk sebuah hutan yang terdiri atas k pohon.

Error klasifikasi *Random Forest* diduga melalui *error* OOB yang diperoleh dengan cara [7]:

1. Lakukan prediksi terhadap setiap data OOB pada pohon yang bersesuaian. Data OOB (*out of bag*) adalah data yang tidak termuat dalam contoh *bootstrap*.
2. Secara rata-rata, setiap amatan gugus data asli akan menjadi data OOB sebanyak sekitar 36% dari banyak pohon. Oleh karena itu, pada langkah 1, masing-masing amatan gugus data asli mengalami prediksi sebanyak sekitar sepertiga kali dari banyaknya pohon. Jika a adalah sebuah amatan dari gugus data asli, maka hasil prediksi *Random Forest* terhadap a adalah gabungan dari hasil prediksi setiap kali a menjadi data OOB.
3. *Error* OOB dihitung dari proporsi misklasifikasi hasil prediksi *Random Forest* dari seluruh amatan gugus data asli.

Algoritma *Naive Bayes*

Algoritma *Naive Bayes* dikemukakan oleh salah satu ilmuwan dari Inggris Thomas Bayes dengan memprediksi probabilitas di masa depan berdasarkan pengalaman di masa sebelumnya. *Naive Bayes* merupakan teknik *data mining* dengan pendekatan teori probabilitas untuk membangun sebuah model klasifikasi berdasarkan pada kejadian masa lalu yang mempunyai potensi membentuk sebuah objek baru yang dikategorikan

sebagai kelas yang memiliki probabilitas terbaik [9]. *Bayes* menggunakan asumsi yang sangat kuat pada independensi antara *predictor*. Bentuk umum Teorema *Bayes* sebagai berikut:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

$P(H|X)$ = Probabilitas hipotesis H berdasarkan kondisi X (prosteriori probabilitas)

$P(X|H)$ = Probabilitas X berdasarkan kondisi tersebut

$P(H)$ = Probabilitas hipotesis H (prior probabilitas)

$P(X)$ = Probabilitas dari X

X = Data dengan *class* yang belum diketahui

H = Hipotesis dari X merupakan *class* spesifik

Untuk diketahui dari rumus tersebut kejadian H mewakilkan sebuah kelas dan X mewakilkan sebuah atribut.

Algoritma k-*Nearest Neighbor*

Fungsi jarak *Euclidean* didefinisikan sebagai berikut:

$$d(X_i - X_j) = \sqrt{\sum_{k=1}^{p} (X_{i,k} - X_{j,k})^2}$$

d = Jarak *Euclidean*

p = Jumlah *predictor*

X_i = Hasil pengamatan pertama

X_j = Hasil pengamatan kedua/berikutnya

Algoritma *Rule Based*

Merupakan algoritma yang bekerja berdasarkan aturan (*rule*) yang telah ditetapkan sebelumnya. Keunggulan dari algoritma ini adalah efesiennya dalam memproses *dataset* yang besar dan *noisy*. Aturan klasifikasi dapat dinyatakan dengan cara berikut:

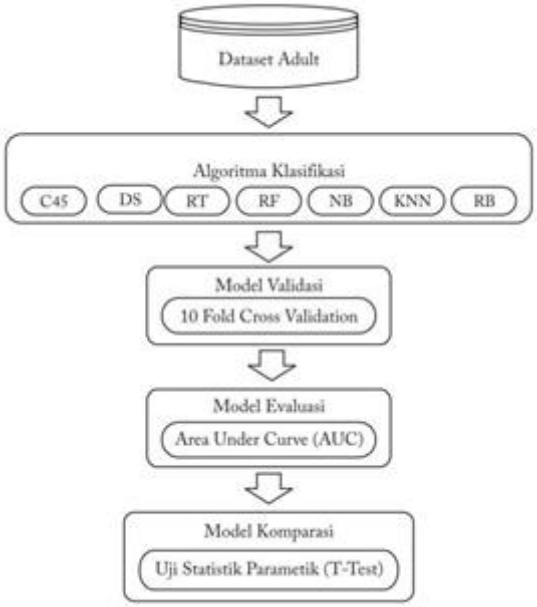
Aturan: (kondisi) $\rightarrow$ y

Di mana kondisi adalah konjungsi atribut-atribut, sedangkan y adalah label kelas. Sisi kiri dari aturan disebut prasyarat yang berisi gabungan dari *test* atribut:

Condition = (A1 op v1), (A2 op v2), ..., (Ak op vk), dimana (Aj, vj) adalah pasangan atribut nilai dan op adalah operator logis yang dipilih dari set {=, =, <, >, ≤, ≥}. Setiap *test* atribut (Aj, op v) dikenal sebagai diperbantukan. Sisi kanan dari aturan disebut *rule consequent*, yang berisi kelas yang diprediksi yi (Tan, dkk, 2006).

METODE PENELITIAN

Dari *dataset* tersebut penulis mencoba membandingkan dengan tujuh algoritma klasifikasi, kemudian dengan metode validasi, evaluasi, menggunakan metode perbandingan uji beda parametik t-*test* untuk membandingkan akurasi algoritma klasifikasi.



Gambar 1. Struktur Metode yang Diusulkan

Dataset

Dalam melakukan penelitian ini penulis mengambil *dataset* dari *web* yang telah di*publish*, dari UCI *machine learning repository*, tentang data kependudukan. Untuk mendapatkan *dataset* tersebut penulis mengambilnya dari alamat https://archive.ics.uci.edu/ml/datasets/Adult.

Algoritma Klasifikasi

Pada penelitian ini penulis menggunakan tujuh algoritma klasifikasi untuk memprediksi pendapatan yang melebihi Rp. 50.000 dan yang kurang atau sama dengan Rp. 50.000, antara lain algoritma *Decission Tree* (C4.5), algoritma *Decission Stump*, algoritma *Random Tree*, algoritma *Random Forest*, algoritma *Naive Bayes*, algoritma k-NN, algoritma *Rule Based*/*Rule Induction*.

Metode Validasi

Metode validasi yang digunakan peneliti adalah k-*Fold Cross Validation* dimana K sama dengan 10 artinya *dataset* dibagi menjadi sepuluh bagian dimana salah satu bagian dijadikan sebagai *data testing* dan yang lain sebagai *data training*. Proses validasi dilakukan berulang-ulang sebanyak sepuluh kali dimana dilakukan *data testing* yang berbeda-beda. Metode ini sering digunakan peneliti sebelumnya sebagai metode standar validasi data. Metode ini dapat dilihat pada tabel 1.

Tabel 1. *Stratified* 10 *Cross Validation*

Testing	***Dataset***									
1	■									
2		■								
3			■							
4				■						
5					■					
6						■				
7							■			
8								■		
9									■	
10										■

Metode Evaluasi

Sebagai landasan mengetahui atau indikator tingkat akurasi performasi dari algoritma klasifikasi dengan menggunakan metode evaluasi untuk menentukan nilai akurasi dari *confusion matrix* dan nilai *Area Under Curve* (AUC) dari ROC *curve*. Nilai akurasi diperoleh dari tabel *confusion matrix* yang diperoleh hasil dari penelitian menggunakan *machine learning Rapid Miner*. *Accuracy* mengacu pada pengukuran tingkat keakuratan atau prediksi dari suatu model atau metode klasifikasi [10]. Adapun nilai tingkat keakuratan klasifikasi dengan menggunakan AUC ditunjukan pada tabel 2.

Tabel 2. Klasifikasi AUC [11]

Performance	**Klasifikasi**
0.90 – 1.00	Paling baik
0.80 – 0.90	Baik
0.70 – 0.80	Adil atau sama
0.60 – 0.70	Rendah
0.50 – 0,60	Gagal

Metode Perbandingan

Untuk mengetahui akurasi algoritma klasifikasi antara satu dengan yang lainnya, penulis menggunakan uji beda parametik t-*test*. Dengan menggunakan t-*test* penulis dapat membandingkan nilai akurasi yang diperoleh antara algoritma satu dengan algoritma yang lain untuk memastikan apakah ada perbedaan signifikan pada akurasi algoritma tersebut. Jika perbedaan antara dua rata-rata akurasi tidak signifikan, dapat dikatakan bahwa akurasi algoritma tidak dapat dibedakan dan jika perbedaannya signifikan, maka salah satu algoritma memiliki akurasi yang tidak bagus dibandingkan algoritma yang lain [12].

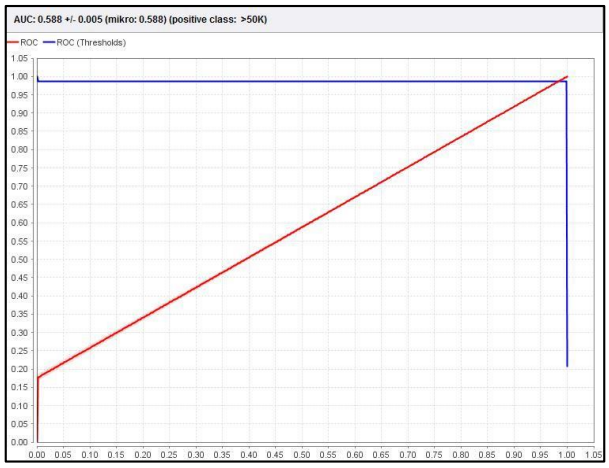
PEMBAHASAN DAN HASIL

Untuk melakuakan validasi, evaluasi, dan perbandingan pada tujuh algoritma klasifikasi tersebut penulis menggunakan aplikasi *Rapid Miner* didukung perangkat keras komputer dengan spesifikasi mengunakan *processor Intel Core* i5 2.35 GHz CPU, 3 GB RAM dan sistem operasi *Microsoft Windows* 8 *Enterprise* 32 *bit*. Hasil evaluasi performansi dari tujuh algoritma klasifikasi dideskripsikan dalam tabel pengukuran akurasi *confusion matrix* dan AUC pada tiap-tiap algoritma tersebut.

Gambar 2. Hasil Grafik AUC Algoritma C4.5

Tabel 3. Hasil *Accuracy* Algoritma *Decision Tree* (C4.5)

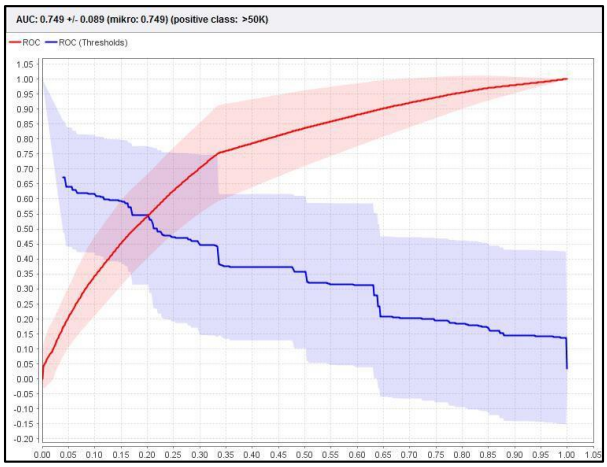
Accuracy: 82.01% +/- 0.32% (mikro 82.01%)			
	True <=50K	*True* >50K	*Class precision*
Pred <=50K	24622	5761	81.04%
Pred >50K	97	2080	95.54%
Class recall	99.50%	26.53%	



Gambar 3. Hasil Grafik AUC Algoritma *Decision Stump*

Tabel 4. Hasil *Accuracy* Algoritma *Decision Stump*

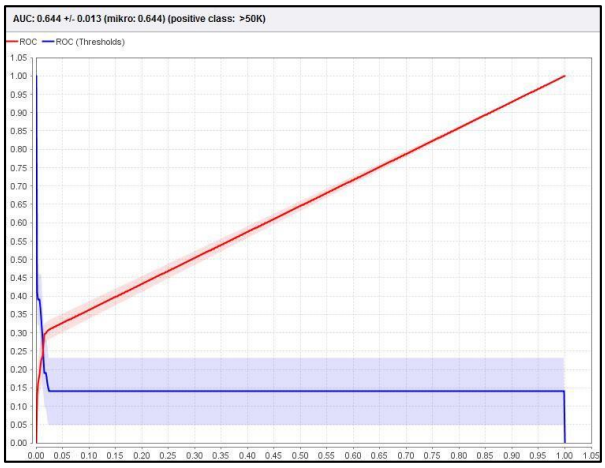
Accuracy: 80.09% +/- 0.26% (mikro 80.09%)			
	True <=50K	*True* >50K	*Class precision*
Pred <=50K	24699	64.62	79.26%
Pred >50K	20	1379	98.57%
Class recall	99.92%	17.59%	



Gambar 4. Hasil Grafik AUC Algoritma *Random Tree*

Tabel 5. Hasil *Accuracy* Algoritma *Random Tree*

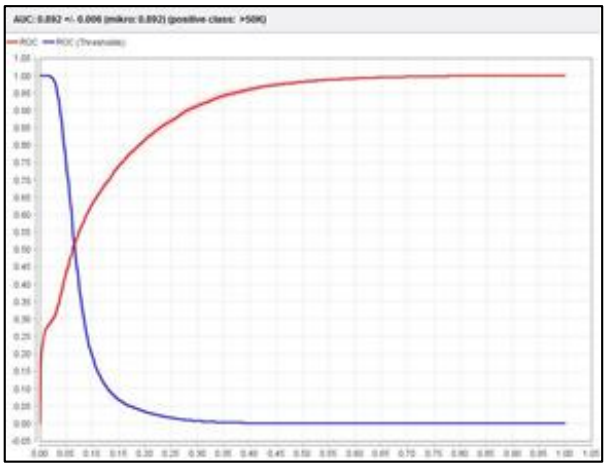
Accuracy: 78.38% +/- 2.62% (mikro 78.38%)			
	True <=50K	*True* >50K	*Class precision*
Pred <=50K	24050	6369	79.06%
Pred >50K	669	1472	68.75%
Class recall	97.29%	18.77%	



Gambar 5. Hasil Grafik AUC Algoritma *Random Forest*

Tabel 6. Hasil *Accuracy* Algoritma *Random Forest*

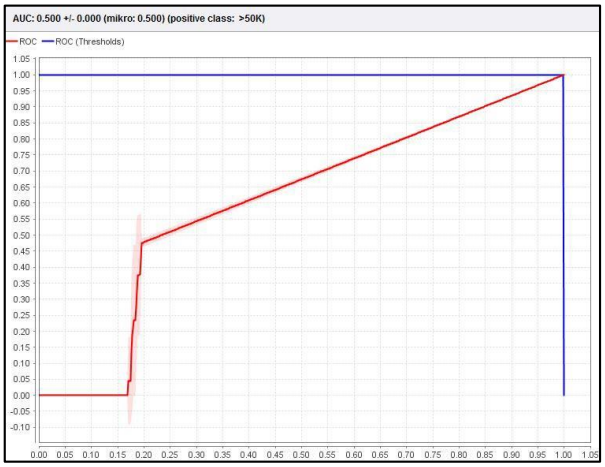
Accuracy: 76.56% +/- 0.82% (mikro 76.56%)			
	True <=50K	*True* >50K	*Class precision*
Pred <=50K	24718	7632	76.41%
Pred >50K	1	209	99.52%
Class recall	100%	2.67%	



Gambar 6. Hasil Grafik AUC Algoritma *Naive Bayes*

Tabel 7. Hasil *Accuracy* Algoritma *Naive Bayes*

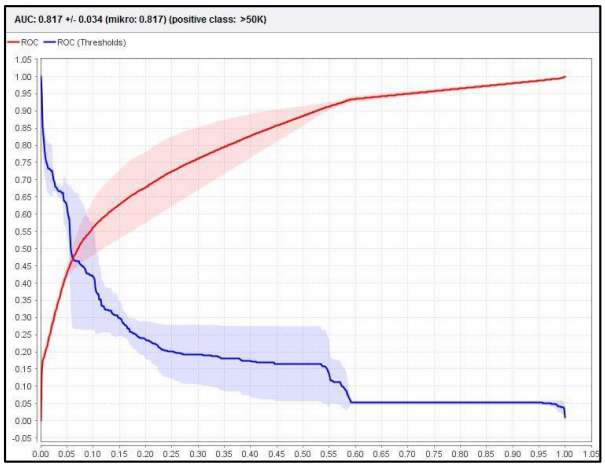
Accuracy: 83.34% +/- 0.44% (mikro 83.34%)			
	True <=50K	*True* >50K	*Class precision*
Pred <=50K	23065	3770	85.95%
Pred >50K	1654	4071	71.11%
Class recall	93.31%	50.92%	



Gambar 7. Hasil Grafik AUC Algoritma k-*Nearest Network*

Tabel 8. Hasil *Accuracy* Algoritma k-*Nearest Network*

Accuracy: 73.32% +/- 0.58% (mikro 73.32%)			
	True <=50K	*True* >50K	*Class precision*
Pred <=50K	20219	4187	82.84%
Pred >50K	4500	3654	44.81%
Class recall	81.80%	46.60%	



Gambar 8. Hasil Grafik AUC Algoritma *Rule Based*

Tabel 9. Hasil *Accuracy* Algoritma *Rule Based*

Accuracy: 83.08% +/- 0.67% (mikro 83.08%)			
	True <=50K	*True* >50K	*Class precision*
Pred <=50K	22820	3610	86.34%
Pred >50K	1899	4231	69.02%
Class recall	92.32%	53.96%	

Hasil pengukuran akurasi menggunakan *confusion matrix* dan AUC dari masing-masing algoritma klasifikasi dapat di rangkum pada tabel berikut ini:

Tabel 10. Hasil *Accuracy* dan AUC

Algoritma	***Accuracy***	**AUC**
C4.5	82.21%	0.633
DS	80.09%	0.588
RT	78.38%	0.749
RF	76.56%	0.644
NB	**83.34%**	**0.892**
k-NN	73.32%	0.500
RB	83.08%	0.817

Dengan memperhatikan tabel di atas yang menampilkan hasil perbandingan performasi masing-masing algoritma diperoleh bahwa *Naive Bayes* memiliki nilai akurasi tertinggi yaitu 83.34%, kemudian algoritma *Rule Based* dengan nilai akurasi 83.08%, selanjutnya *Decision Tree* C4.5 82.21%, *Decision Stump* 80.09%, *Random Tree* 78.38%, *Random Forest* 76,56%, k-NN 73.32%. Sedangkan dilihat dari nilai AUC dari tiap-tiap algoritma klasifikasi tersebut algoritma *Naive Bayes* memiliki nilai tertinggi yaitu 0.892, diurutan kedua algoritma *Rule Based* 0.817, urutan selanjutnya algoritma *Random Tree* 0.749, *Random Forest* 0.644, *Decision Tree* C4.5 0.633, *Decision Stump* 0.588, k-NN 0.500. Hasil akurasi dari tujuh algoritma klasifikasi dengan uji beda parametik t-*test* dapat dilihat pada tabel di bawah:

Tabel 11. Uji Beda Parametik T-*Test*

	C4.5	**DS**	**RT**	**RF**	**NB**	**k-NN**	**RB**
C4.5		0.000	0.000	0.000	0.000	0.000	0.001
DS			0.077	0.000	0.000	0.000	0.000
RT				0.001	0.000	0.000	0.000
RF					0.000	0.000	0.000
NB						0.000	0.414
k-NN							0.000
RB							

Dilihat dari tabel 11, berdasarkan uji beda parametik t-*test* menunjukkan bahwa algoritma *Decision Tree* C4.5 mempunyai nilai yang paling baik yang menunjukan adanya beda signifikan terhadap algoritma yang lain yaitu algoritma *Decision Stump*, *Random Tree*, *Random Forest*, *Naive Bayes*, k-*Nearest Neighbor*, dan *Rule Based*.

KESIMPULAN

Dari hasil komparasi tujuh algoritma klasifikasi pada *dataset adult* melalui uji statistik parametrik *(Paired Sample* T-*Test*), bahwa algoritma *Naive Bayes* memiliki nilai akurasi tertinggi yaitu 83.34%, sedangkan dilihat dari nilai AUC dari tiap-tiap algoritma klasifikasi tersebut algoritma *Naive Bayes* memiliki nilai tertinggi yaitu 0.892, berdasarkan uji beda parametik t-*test* memperlihatkan bahwa algoritma *Decision Tree*

C4.5 memiliki nilai dimana menunjukkan adanya beda signifikan terhadap algoritma yang lain yaitu algoritma *Decision Stump*, *Random Tree*, *Random Forest*, *Naive Bayes*, k-*Nearest Neighbor*, dan *Rule Based*.

DAFTAR PUSTAKA

[1] M. F. bin Othman and T. M. S. Yau, "Comparison of Different Classification Techniques using WEKA for Breast Cancer," *FMBE Proc.*, vol. 15, pp. 520–523, 2007.

[2] R. S. Wahono, N. S. Herman, and S. Ahmad, "A Comparison Framework of Classification Models for Software Defect Prediction," vol. 20, no. 10, pp. 1945–1950, 2014, doi: 10.1166/asl.2014.5640.

[3] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining "Practical Machine Learning Tools and Techniques,"* Third Edit. Burlington: Morgan Kaufmann Publishers, 2011.

[4] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining Third Edition*. Elsevier Inc., 2011.

[5] R. Kohavi and R. Quinlan, "Decision Tree Discovery," 1999.

[6] F. Akthar and C. Hahne, *Rapidminer 5 "Operator Reference."* 2012.

[7] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[8] L. Torgo, *Data Mining with R: Learning with Case Studies*. English: Chapman and Hall, 2010.

[9] D. Farid, L. Zhang, C. Mofizur, M. A. Hossain, and R. Strachan, "Expert Systems with Applications Hybrid Decision Tree and Naive Bayes Classifiers for Multiclass Classification Tasks," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1937–1946, 2014, doi: 10.1016/j.eswa.2013.08.089.

[10] C. Sammut, *Encyclopedia of Machine Learning*. Springer, 2011.

[11] F. Gorunescu, *Data Mining Concepts, Models, and Techniques*. Verlag Berlin Heidelberg: Springer, 2011.

[12] H. Crc and M. Hofmann, *RapidMiner: Data Mining use Cases and Business Analytics Applications*. CRC Press, 2014.