IJAI (Indonesian Journal of Applied Informatics)
Vol. 10 No. 1 Tahun 2025 pISSN: 2548-3846, eISSN: 2598-5981

Analisis Perbandingan Algoritma *Decision Tree* dengan *Random Forest* dalam Deteksi *Bot DDOS*

Kristianto Pratama Dessan Putra [1*], Rianto Rianto[1], EIH Ujianto[1]
[1] Magister Teknologi Informasi, Universitas Teknologi Yogyakarta
Email: kristianto0704@gmail.com

Info Artikel

Kata Kunci :
bot ddos, decision tree, random forest

Keywords :
bot ddos, decision tree, random forest

Tanggal Artikel
Dikirim : 30 Mei 2025
Direvisi : 24 Desember 2025
Diterima : 30 Desember 2025

Abstrak

Tingkat penetrasi internet yang semakin meningkat setiap tahunnya juga berpengaruh pada banyaknya peralihan layanan dari konvensional ke *platform* internet. Peralihan layanan tersebut terbukti membawa dampak baik, seperti meningkatnya *volume* penjualan produk. Namun, di sisi lain dengan semakin banyaknya peralihan layanan ke *platform* internet maka semakin banyak pula celah-celah keamanan yang dapat dieksploitasi, salah satunya serangan bot DDos. Oleh karena itu, diperlukan adanya sistem yang mampu mendeteksi serangan bot DDos dan algoritma yang akan dianalisis dalam penelitian ini adalah *Decision Tree* dan *Random Forest*. Penelitian ini akan membandingkan kedua algoritma tersebut untuk menentukan algoritma yang paling optimal dalam mendeteksi serangan bot DDos. Penelitian ini menggunakan dua dataset dalam proses implementasi algoritma, yaitu KDD CUP 1999 dan CICIDS 2017. Ruang lingkup dari perbandingan kedua algoritma meliputi tingkat akurasi dan durasi waktu pemrosesan data. Hasil dari penelitian menunjukkan bahwa algoritma *Random Forest* unggul tipis dalam hal tingkat akurasi dibandingkan dengan *Decision Tree*, yaitu 0.9998 untuk *Random Forest* berbanding 0.9997 untuk *Decision Tree*. Namun, algoritma *Decision Tree* unggul jauh dalam hal durasi waktu dibandingkan dengan *Random Forest*, yaitu 20-30 detik untuk *Decision Tree* berbanding 210-300 detik untuk *Random Forest*. Hal tersebut dapat terjadi dikarenakan *Random Forest* memproses lebih banyak pohon kemungkinan dibandingkan *Decision Tree*.

Abstract

The increasing internet penetration each year also affects the shift of services from conventional methods to internet platforms. This shift has proven to bring positive impacts, such as an increase in product sales volume. However, there are increasingly more security vulnerabilities that can be exploited, such as DDoS bot attacks. Therefore, a system that capable to detect bot DDoS attacks is needed. This study compares these two algorithms (Decision Tree and Random Forest) to determine which is the most optimal for detecting bot DDoS attacks. The scope of the comparison includes accuracy levels and data processing time. The results show that Random Forest slightly outperforms Decision Tree in terms of accuracy, with a score of 0.9998 for Random Forest compared to 0.9997 for Decision Tree. However, Decision Tree is significantly superior in processing time compared to Random Forest (20–30 seconds for Decision Tree versus 210–300 seconds for Random Forest). This occurs because Random Forest processes more trees than Decision Tree.

1. PENDAHULUAN

Perkembangan internet di Indonesia terus mengalami peningkatan jumlah setiap tahunnya. Hal ini dibuktikan dengan jumlah penetrasi internet yang terus mengalami pertumbuhan setiap tahunnya. Berdasarkan data dari APJII (Asosiasi Penyelenggara Jasa Internet Indonesia), penetrasi internet di Indonesia terus mengalami peningkatan mulai dari tahun 2018 dengan persentase 64.8% hingga tahun 2023 dengan persentase 78.19%. Bahkan, diprediksi tahun 2024 akan kembali mengalami peningkatan dengan persentase 79%. Pertumbuhan ini tentu menarik minat dari berbagai penyedia layanan, baik jasa ataupun barang, untuk menyediakan layanannya secara *online* di internet [1]. Berbagai sektor, mulai dari retail, jasa, hingga teknologi, banyak menawarkan layanannya pada *platform* internet.

Peralihan layanan ke *platform* internet memang memungkinkan penetrasi produk mengalami peningkatan karena dapat diakses kapanpun dan dimanapun, bahkan dapat meningkatkan volume penjualan [2]. Namun, di sisi lain semakin banyak celah-celah keamananan yang dapat dieksploitasi seiring dengan pertumbuhan layanan di *platform* internet [3]. Berbagai bentuk serangan, seperti *ransomware*, serangan *brute force, botnets, man-in-the-middle attack*, hingga DDos dapat terjadi pada layanan yang ada di *platform* internet [4]. Hal ini tentunya memberikan dampak negatif apabila celah-celah keamanan tersebut dapat dieksploitasi oleh pihak-pihak yang tidak bertanggung-jawab.

Salah satu bentuk eksploitasi keamanan yang sering terjadi pada layanan di *platform* internet adalah serangan Bot DDos. Cara kerja dari serangan Bot DDos adalah membanjiri *traffic* dari suatu layanan, contohnya *website* ataupun aplikasi, dengan *request-request* yang menyebabkan *user-user* sah tidak dapat mengakses layanan tersebut [5]. Dampak lebih lanjut dari serangan Bot DDos adalah layanan tersebut mengalami *down*, yaitu kondisi di mana layanan tersebut sama sekali tidak dapat diakses karena telah dipenuhi oleh *request* tidak sah dari Bot DDOs [6]. Jika serangan Bot DDos tidak diantisipasi dengan tepat, maka kerugian dalam bentuk finansial dapat terjadi, seperti penurunan jumlah penjualan hingga kegagalan proses transaksi pada layanan tersebut.

Oleh sebab itu, diperlukan adanya sistem yang mampu mendeteksi serangan Bot DDos seefektif dan seefisien mungkin agar dampak dari serangan Bot DDos tersebut dapat diminimalisir [7]. Ada berbagai algoritma *Machine Learning* yang dapat digunakan untuk mengklasifikasikan serangan Bot DDos, diantaranya *Random Forest, Decision Tree, Support Vector Machine, Naive Bayes*, KKN, dan lain-lain. Algoritma-algoritma tersebut mampu mengklasifikasikan *user-user* yang mengakses layanan internet dan kemudian mendeteksi *user-user* mana saja yang terindikasi sebagai Bot DDos [8]. Dengan menerapkan salah satu dari algoritma tersebut maka sistem dapat melakukan pencegahan sedini mungkin terhadap serangan Bot DDOs pada layanan tersebut.

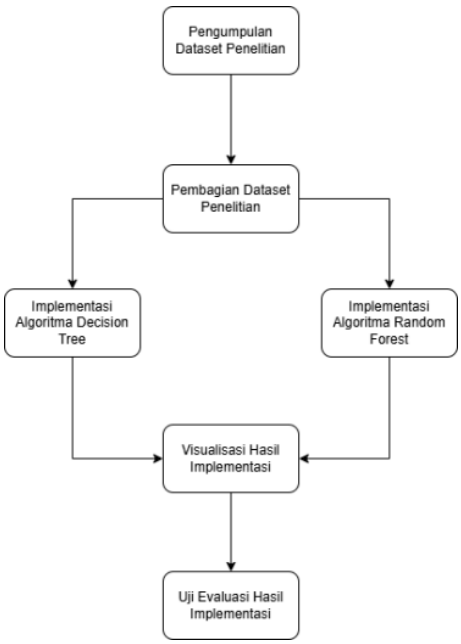
Dalam penelitian yang dilakukan oleh [9] algoritma *Decision Tree* dibandingkan dengan SVM dan *Naive Bayes* dan didapatkan hasil bahwa *Decision Tree* menjadi algoritma dengan tingkat akurasi tertinggi, yaitu 99%. Lebih lanjut, dalam penelitian yang dilakukan oleh [7] menunjukan bahwa *Random Forest* menjadi algoritma dengan tingkat akurasi tertinggi saat dibandingkan dengan *Decision Tree* dan SVM. Dan lagi, dalam penelitian yang dilakukan oleh [8] membuktikan bahwa *Random Forest* menjadi algoritma dengan tingkat akurasi tertinggi, yaitu 99.41% saat dibandingkan dengan SVM, KKN, dan MLP.

Penelitian ini akan membandingkan efektifitas dan efisiensi dari algoritma *Random Forest* dan *Decision Tree* dalam mendeteksi serangan Bot DDos. Alasan dari pemilihan algoritma tersebut adalah algoritma tersebut banyak terbukti menjadi algoritma dengan tingkat akurasi tertinggi. Diharapkan hasil dari penelitian ini dapat memberikan referensi algoritma terbaik untuk mengklasifikasikan dan mendeteksi serangan Bot Ddos.

2. METODE PENELITIAN

Penelitian ini dilakukan dengan menggunakan data sekunder yang terdapat pada situs kaggle yaitu dataset A dan B. Tujuan dari penggunaan dua dataset adalah untuk memvalidasi hasil dari penelitian agar tidak bergantung hanya pada salah satu dataset. Selanjutnya, dataset akan diproses dengan dengan algoritma *Decision Tree* dan *Random Forest* untuk menghitung tingkat akurasi dari kedua model algoritma tersebut.

Tahapan metode dari penelitian ini tersaji pada Gambar 1.



Gambar 1. Metode Penelitian

2.1 Pengumpulan Dataset Penelitian

Pada penelitian ini, ada dua dataset yang akan digunakan, dataset yang digunakan berupa data sekunder yang didapatkan dari situs kaggle. Dataset yang akan digunakan pada penelitian ini merupakan jenis dataset klasifikasi yang memiliki kumpulan baris data dengan berbagai jenis kategori. Alasan dari pemilihan dua dataset dalam penelitian ini adalah untuk memvalidasi bahwa salah satu algoritma unggul dalam pemrosesan sekalipun berbeda dataset. Selain itu, penggunaan dua dataset juga ditujukan untuk mengukur waktu yang dibutuhkan oleh masing-masing algoritma untuk melakukan proses klasifikasi. Masing-masing dataset tersebut harus terdiri dari ratusan ribu baris data agar dapat merepresentasikan serangan dari Bot DDos. Selanjutnya, dataset tersebut akan di-*import* ke dalam *python* untuk kemudian diproses lebih lanjut. Proses *import* perlu dipastikan berhasil agar tahapan selanjutnya dapat dijalankan.

2.2 Pembagian Dataset Penelitian

Setelah dataset telah sukses di-*import*, maka tahapan selanjutnya adalah membagi dataset penelitian menjadi 2 bagian, yaitu data *training* dan data *testing*. Semakin banyak data training yang digunakan akan berdampak pada peningkatan akurasi hasil algoritma [10]. Oleh karena itu, dataset yang digunakan untuk *training* dipilih sebesar 80% dan untuk *testing* sebesar 20% sehingga komposisi masing-masing untuk kedua dataset adalah sebagai berikut.

- KDD CUP 1999 : 395.217 (*Training*) dan 98.804 (*Testing*)
- CICIDS 2017 : 180.596 (*Training*) dan 45.149 (*Testing*)

Selanjutnya, data *testing* akan digunakan untuk memvalidasi hasil dari algoritma, bertujuan untuk menguji tingkat akurasi dari masing-masing algoritma.

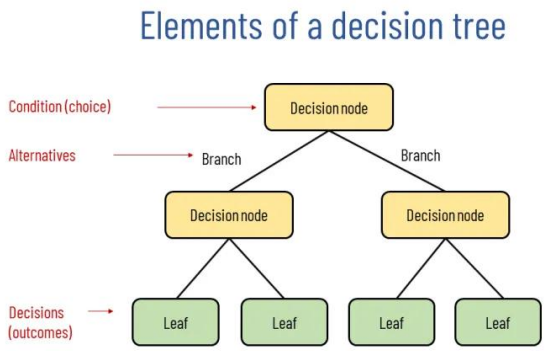
2.3 Implementasi Algoritma *Decision Tree* dan *Random Forest*

Pada tahapan ini, dataset yang sudah dibagi akan lanjut diproses dengan memodelkannya ke dalam 2 bentuk algoritma, yaitu *Decision Tree* dan *Random Forest*. Kedua algoritma tersebut akan membangun pohon-pohon keputusan untuk mempelajari pola dan perilaku dari setiap data, apabila ditemukan kesamaan karakter atau perilaku maka akan dikelompokan ke dalam kategori yang sama. Hal tersebut yang menjadi alasan mengapa *Decision Tree* dan *Random Forest* menjadi pilihan yang tepat dalam hal klasifikasi. Hasil akhir dari tahapan ini adalah setiap dataset berhasil diproses oleh masing-masing algoritma dan didapatkan hasil akurasi dalam mendeteksi serangan Bot Ddos.

2.3.1 Decision Tree

Decision Tree merupakan salah satu algoritma (*Machine Learning*) klasifikasi yang bekerja dengan cara membangun pohon-pohon keputusan dalam bentuk cabang-cabang, setiap pohon keputusan mewakili nilai atau *value* yang memiliki kemungkinan terjadi [11]. *Decision Tree* akan memproses dataset ke dalam kelompok-kelompok kecil berdasarkan atribut yang dimiliki oleh data tersebut sehingga hasil akhirnya adalah terbentuk kelas-kelas data dengan kesamaan atribut [12].

Lebih lanjut, *Decision Tree* terbagi atas 2 bagian, yaitu *decision node* dan *leaf node*, *decision node* berperan menghasilkan *leaf node* yang lebih kecil dan lebih kecil lagi dengan suatu variabel pembeda [13] seperti yang tersaji pada Gambar 2. Algoritma *Decision Tree* dipilih sebagai algoritma perbandingan dalam penelitian ini karena mampu menjadi algoritma dengan tingkat akurasi tertinggi saat dibandingkan dengan algoritma lainnya pada beberapa artikel penelitian sebelumnya, salah satu sebabnya adalah karena *Decision Tree* mampu menghasilkan lebih banyak kelas-kelas data daripada algoritma lainnya [14]. Selain itu, waktu eksekusi dari *Decision Tree* juga lebih cepat daripada algoritma lainnya, sehingga deteksi Bot DDos dapat dilakukan sesegera mungkin [15].

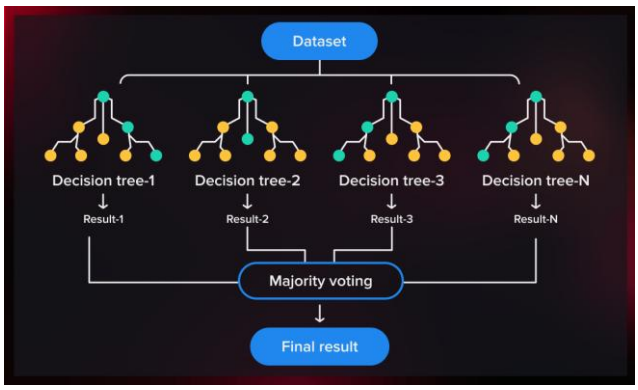


Gambar 2. Decision Tree

2.3.2 Random Forest

Random Forest merupakan salah satu algoritma (*Machine Learning*) klasifikasi yang memiliki cara kerja hampir sama dengan *Decision Tree*, yaitu membangun pohon-pohon keputusan. Perbedaan utama dengan *Decision Tree* adalah *Random Forest* membangun lebih banyak pohon-pohon keputusan, dapat disebut bahwa *Random Forest* merupakan kumpulan set *Decision Tree* [16] seperti yang tersaji pada Gambar 3. *Random Forest* merupakan algoritma yang cocok diterapkan pada dataset yang berukuran besar karena mampu membangun lebih dari satu *Decision Tree* dan mengelompokan tiap-tiap data berdasarkan klasifikasinya [17], semakin banyak pohon keputusan yang dihasilkan maka semakin tinggi tingkat akurasi hasilnya [10].

Algoritma *Random Forest* dipilih sebagai algoritma perbandingan dalam penelitian ini karena juga mampu menjadi algoritma dengan tingkat akurasi tertinggi saat dibandingkan dengan algoritma lainnya pada beberapa beberapa artikel penelitian sebelumnya, salah satu alasannya karena *Random Forest* mampu menghasilkan pohon-pohon kemungkinan dalam jumlah banyak sehingga kelas-kelas data yang dihasilkan menjadi lebih banyak [18]. Namun, waktu pemrosesan untuk *Random Forest* lebih lama dibandingkan dengan algoritma lainnya, ini berkaitan dengan jumlah pohon keputusan yang lebih banyak pada algoritma *Random Forest* [11]. Implementasi dataset ke dalam algoritma *Random Forest* akan dilakukan dengan *Python* dan memanfaatkan beberapa *library*, yaitu *numpy*, *matlab*, dan *sklearn* [8].



Gambar 3. Random Forest

2.4 Visualisasi Hasil Implementasi

Tahapan selanjutnya adalah visualisasi hasil implementasi ke dalam bentuk grafik. Setiap tahapan, mulai dari *import,* pemrosesan dataset, hingga hasil implementasi, akan divisualisasikan guna mempermudah peninjauan hasil.

2.5 Uji Evaluasi Hasil Implementasi

Tahapan uji evaluasi dari hasil implementasi akan dibantu dengan *library* dari *python*, yaitu *sklearn metrics*. Hasil akhir dari tahapan uji evaluasi adalah perbandingan antara *Decision Tree* dengan *Random Forest* pada beberapa matriks, diantaranya *accuracy, precision, recall,* dan *F1- score* [19]. Selain itu, uji evaluasi juga akan melibatkan *cross-validation* untuk memastikan hasil akhir dari implementasi algoritma [20].

3. HASIL DAN PEMBAHASAN

Hasil dari setiap tahapan alur penelitian akan dijabarkan pada bagian ini, mulai dari tahapan pemilihan dataset, pembagian dataset, implementasi algoritma, visualisasi hasil, dan uji evaluasi.

3.1 Pemilihan Dataset Penelitian

Dataset yang akan digunakan pada penelitian ini adalah dataset KDD Cup 1999 dan dataset CICIDS 2017. Kedua dataset tersebut dipilih karena mampu merepresentasikan simulasi serang dari Bot Ddos. Selanjutnya, kedua dataset tersebut akan di-*import* ke dalam *Python*. Pengecekan hasil *import* akan diuji dengan menampilkan 5 baris data teratas.

Berikut adalah hasil *import* untuk dataset **KDD CUP 1999**. Informasi dataset KDD CUP 1999 yang meliputi jumlah data dan jumlah kolom tersaji pada Gambar 4. Contoh data dari dataset KDD CUP 1999 tersaji pada Gambar 5.

```
Informasi dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 494021 entries, 0 to 494020
Data columns (total 42 columns):
```

Gambar 4. Dataset KDD CUP 1999

```
Lima baris pertama dataset:
   duration protocol_type service flag  src_bytes  dst_bytes  land  \
0         0           tcp    http   SF        181       5450     0
1         0           tcp    http   SF        239        486     0
2         0           tcp    http   SF        235       1337     0
3         0           tcp    http   SF        219       1337     0
4         0           tcp    http   SF        217       2032     0
```

Gambar 5. Sample Dataset KDD CUP 1999

Berikut adalah hasil import untuk dataset **CICIDS 2017.** Informasi dataset CICIDS 2017 yang meliputi jumlah data dan jumlah kolom tersaji pada Gambar 6. Contoh data dari dataset CICIDS 2017 tersaji pada Gambar 7.

```
Informasi dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225745 entries, 0 to 225744
Data columns (total 79 columns):
```

Gambar 6. Dataset CICIDS 2017

```
Lima baris pertama dataset:
    Destination Port   Flow Duration   Total Fwd Packets  \
0              54865               3                   2
1              55054             109                   1
2              55055              52                   1
3              46236              34                   1
4              54863               3                   2
```

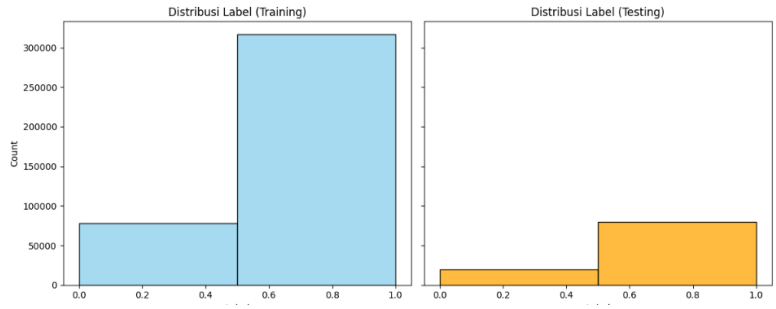
Gambar 7. Sample Dataset CICIDS 2017

3.2 Pembagian Dataset Penelitian

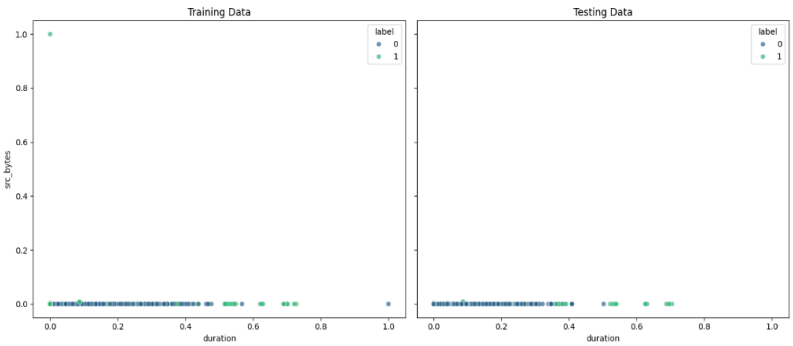
Tahapan selanjutnya adalah membagi dataset penelitian menjadi data *training* dan data *testing*. Persentase pembagian dataset adalah data *training* sebesar 80% dan data *testing* sebesar 20%. Distribusi data *training* dan *testing* pada dataset KDD CUP 1999 tersaji pada Gambar 8 dan sebaran data yang tersaji pada Gambar 9. Lalu, distribusi data training dan testing pada dataset CICIDS 2017 tersaji pada Gambar 10 dan sebaran data yang tersaji pada Gambar 11.

Hasil dari tahapan ini akan divisualisasikan menggunakan *Python* guna menggambarkan peta sebaran data antara data *training* dengan data *testing*. Apabila secara persentase sudah tepat dan secara sebaran juga memiliki kemiripan maka proses ini dianggap sukses. Dan berikut adalah hasil visualisasinya.

KDD CUP 1999

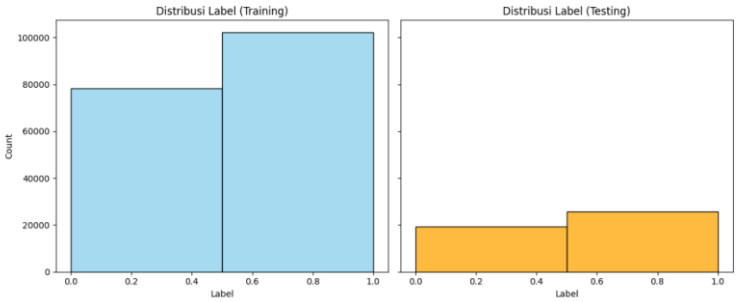


Gambar 8. Distribusi Dataset (Training dan Testing) KDD CUP 1999

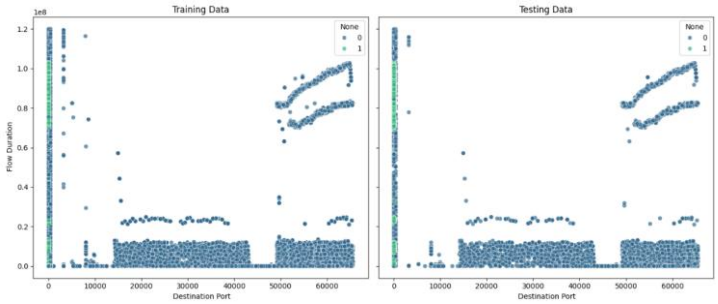


Gambar 9. Sebaran Data (Training dan Testing) KDD CUP 1999

Dataset CICIDS 2017



Gambar 10. Distribusi Dataset (Training dan Testing) CICIDS 2017



Gambar 11. Sebaran Data (Training dan Testing) CICIDS 2017

Dari hasil visualisasi pada Gambar 8 hingga Gambar 11, dapat diperhatikan bahwa peta sebaran data, antara data testing dengan data training, memiliki kemiripan. Dan lagi, persentase distribusi data telah sesuai dengan pembagian di awal, yaitu 80% data training dan 20% data *testing*. Dari kedua hal tersebut, dapat disimpulkan bahwa tahapan pembagian data telah sukses dilakukan

3.3 Implementasi Algoritma

Pada tahapan ini, kedua dataset akan diuji dengan algoritma *Decision Tree* dan *Random Forest* untuk melihat manakah algoritma yang lebih optimal dan lebih akurat dalam memproses dataset tersebut. Fokus utama perbandingan ada pada tingkat akurasi dan durasi waktu yang dibutuhkan untuk pemrosesan dataset pada masing-masing algoritma. Proses implementasi dari setiap algoritma dijalankan pada *coding Python* dengan *library sklearn*. Hasilnya dari proses tersebut tersaji pada Tabel 1 yang mencakup informasi akurasi, *training time*, dan *testing time* dalam pengujian dataset KDD CUP 1999 dan Tabel 2 yang mencakup informasi akurasi, *training time*, dan *testing time* dalam pengujian dataset CICIDS 2017. Tabel 1 yang memproses dataset KDD CUP 1999 menunjukan bahwa algoritma *Random Forest* unggul dalam hal akurasi, yaitu 0.9996 berbanding 0.9995 dari *Decision Tree.* Namun, algoritma *Decision Tree* unggul jauh dalam hal waktu pemrosesan, baik *training time* maupun *testing time,* dengan selisih waktu hingga 10-15x lebih cepat dibandingkan *Random Forest.* Hasil serupa juga ditunjukan oleh Tabel 2 yang memproses dataset CICIDS 2017 dimana *Random Forest* unggul secara akurasi dan *Decision Tree* unggul secara waktu pemrosesan. Hasil tersebut dipengaruhi oleh perbedaan cara kerja antara *Random Forest* dengan *Decision Tree*, *Random Forest* memproses lebih banyak pohon kemungkinan dibandingkan *Decision Tree* sehingga membutuhkan waktu lebih lama namun memberikan hasil akurasi yang lebih tinggi.

Dataset KDD CUP 1999

Tabel 1. Hasil Implementasi Dataset KDD CUP 1999

Algoritma	Accuracy	Training Time	Testing Time
Decision Tree	0.9995546784069632	6.61592 seconds	0.08494 seconds
Random Forest	0.9996660088052224	70.59551 seconds	1.58203 seconds

Dataset CICIDS 2017

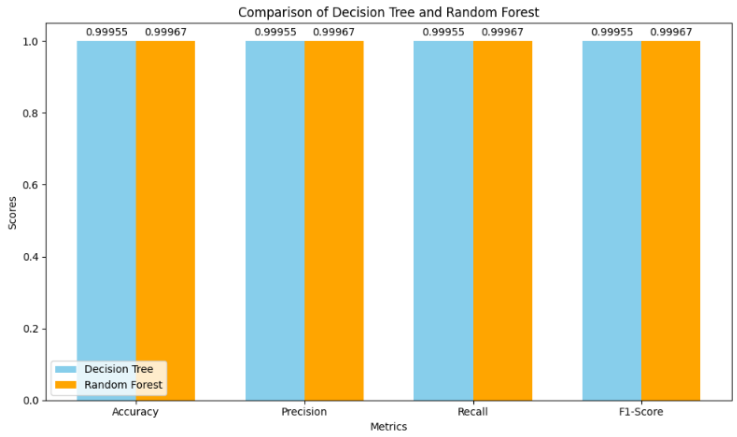
Tabel 2. Hasil Implementasi Dataset CICIDS 2017

Algoritma	Accuracy	Training Time	Testing Time
Decision Tree	0.9997785111519635	5.44173 seconds	0.04997 seconds
Random Forest	0.9998228089215708	81.23361 seconds	0.47171 seconds

3.4 Visualisasi Hasil Implementasi

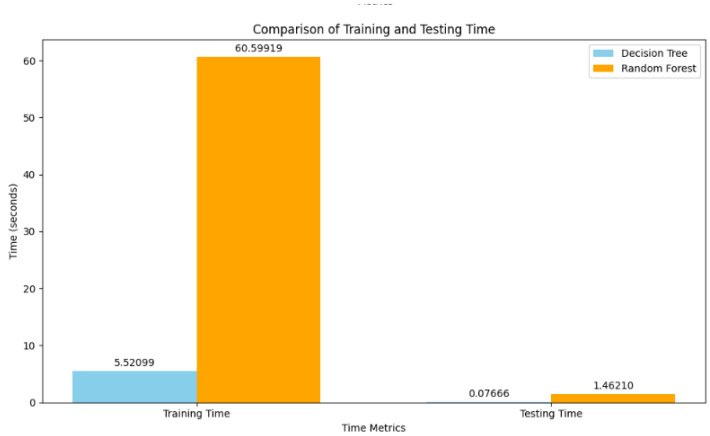
Pada tahapan ini, hasil proses implementasi akan divisualisasikan dalam bentuk grafik perbandingan. Visualisasi bertujuan untuk memudahkan identifikasi perbedaan performa antara kedua algoritma yang diuji dan berikut adalah hasil dari visualisasi tersebut. Gambar 12 menyajikan informasi perbandingan akurasi dan Gambar 13 menyajikan informasi perbandingan waktu proses antara algoritma *Decision Tree* dengan *Random Forest* untuk dataset KDD CUP 1999. Gambar 14 menyajikan informasi perbandingan akurasi dan Gambar 15 menyajikan informasi perbandingan waktu proses antara algoritma *Decision Tree* dengan *Random Forest* untuk dataset CICIDS 2017.

Dataset KDD CUP 1999



Gambar 12. Hasil Perbandingan Akurasi Untuk Dataset KDD CUP 1999

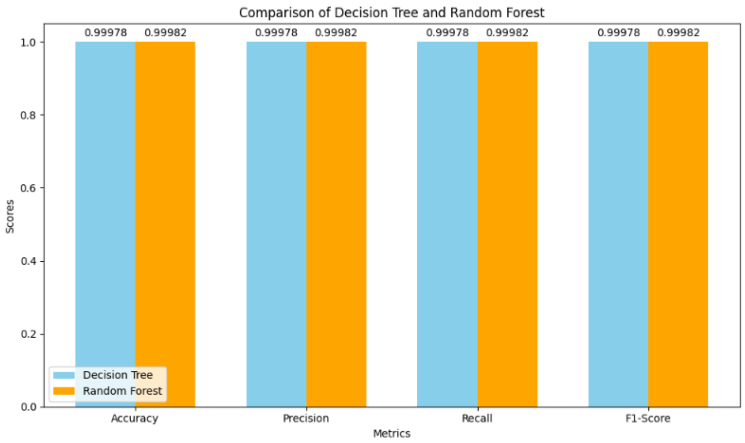
Berdasarkan data beberapa variabel pembanding yang ada pada Gambar 12, yaitu Accuracy, Precision, Recall, dan *F1-Score*, didapatkan hasil bahwa tidak terjadi perbedaan yang signifikan antara *Decision Tree* dengan *Random Forest*. Persentase selisih antara kedua algoritma hanya sebesar 0.0111% dimana algoritma *Random Forest* lebih baik daripada *Decision Tree* dalam memproses dataset KDD CUP 1999.



Gambar 13. Hasil Perbandingan Waktu Untuk Dataset KDD CUP 1999

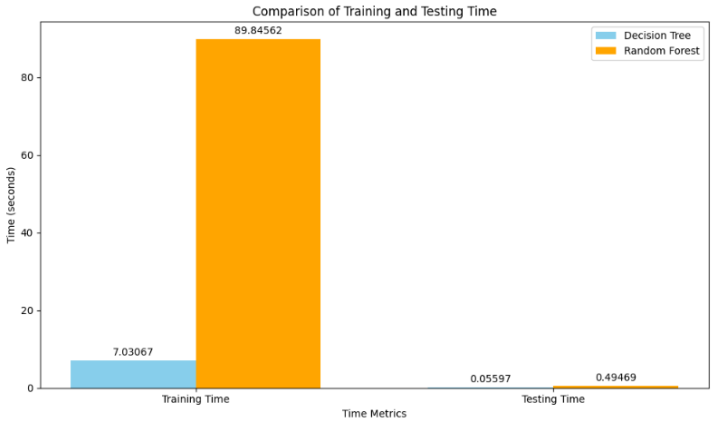
Berdasarkan hasil perbandingan dari variabel *training time* dan *testing time* pada Gambar 13, didapatkan hasil bahwa terjadi perbedaan yang signifikan antara *Decision Tree* dengan *Random Forest*. Persentase perbedaan diantara kedua algoritma tersebut adalah 967.06% untuk *training time* (hampir 10x lipat) dan 1762.53% untuk *testing time* (hampir 20x lipat) algoritma *Decision Tree* lebih cepat memproses dataset daripada *Random Forest.*

Dataset CICIDS 2017



Gambar 14. Hasil Perbandingan Akurasi Untuk Dataset CICIDS 2017

Hasil yang tidak jauh berbeda juga terjadi pada dataset CICIDS 2017 dimana tidak terjadi selisih yang signifikan pada keempat variabel pembanding, yaitu 0.00443% dengan *Random Forest* lebih baik daripada *Decision Tree* seperti yang tersaji pada Gambar 14.



Gambar 15. Hasil Perbandingan Waktu Untuk Dataset CICIDS 2017

Hasil dari perbandingan *training time* dan *testing time* pada dataset CICIDS 2017, yang tersaji pada Gambar 15, juga memberikan hasil yang sama dengan dataset sebelumnya, yaitu *Decision Tree* lebih unggul dalam variabel *training time* dan juga *testing time*. Hasil perbandingan menunjukan bahwa ada perbedaan 1392.49% atau hampir 14x lipat pada *training time* dan 843.99% pada *testing* data dimana *Decision Tree* lebih cepat memproses dataset daripada *Random Forest*. Hal ini bisa terjadi diakibatkan jumlah pohon kemungkinan yang dibangun oleh *Random Forest* jauh lebih banyak daripada *Decision Tree*.

3.5 Uji Evaluasi Hasil Implementasi

Tahapan selanjutnya adalah melakukan uji evaluasi guna memvalidasi hasil dari implementasi algoritma. Apabila ditemukan kesesuaian hasil antara uji evaluasi dengan implementasi algoritma maka proses implementasi dinyatakan sukses. Berikut adalah hasil uji evaluasi dari hasil implementasi algoritma *Decision Tree* dan *Random Forest* yang tersaji pada Tabel 3 dan Tabel 4.

Dataset KDD CUP 1999

Tabel 3. Hasil Uji Evaluasi untuk Dataset KDD CUP 1999

Algoritma	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.999555	0.999555	0.999555	0.999555
Random Forest	0.999666	0.999666	0.999666	0.999666

Decision Tree Cross-Validation
Scores:
[0.98379637 0.99904862 0.9989879 0.99910935 0.99871463]
Decision Tree CV Time: 21.49279 seconds

Random Forest Cross-Validation
Scores:
[0.98443399 0.99991903 0.99994939 0.99979758 0.99976722]
Random Forest CV Time: 214.64646 seconds

Dataset CICIDS 2017

Tabel 4. Hasil Uji Evaluasi untuk Dataset CICIDS 2017

Algoritma	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.999779	0.999779	0.999779	0.999779
Random Forest	0.999823	0.999823	0.999823	0.999823

Decision Tree Cross-Validation
Scores:
[0.99964562 0.99988925 0.9998671 0.99977851 0.99749712]
Decision Tree CV Time: 27.15519 seconds

Random Forest Cross-Validation
Scores:
[0.99988926 0.9999114 0.9998671 0.99997785 0.99920262]
Random Forest CV Time: 305.51827 seconds

Nilai dari uji evaluasi tersebut menunjukan keselarasan dengan hasil implementasi algoritma bahwa *Random Forest* tetap unggul dalam hal akurasi dibandingkan *Decision Tree*, yaitu 0.9998 berbanding 0.9997. Hasil tersebut didukung oleh 3 matriks lainnya yang juga diungguli oleh *Random Forest*, yaitu *Precision, Recall,* dan *F1-Score* dengan selisih sekitar 0.0001. Meskipun selisih yang ditampilkan terlihat tipis namun hasil tersebut tetap menunjukan *Random Forest* lebih unggul. Namun, *Random Forest* tertinggal jauh dalam hal waktu pemrosesan dibandingkan dengan *Decision Tree* dengan selisih waktu hampir 15x lipat. Hal ini telah diprediksi sebelumnya dikarenakan *Random Forest* membangun lebih banyak pohon keputusan dibandingkan *Decision Tree*. Dengan demikian, *Random Forest* dinyatakan unggul secara akurasi namun tertinggal secara waktu pemrosesan terhadap *Decision Tree* pada kedua dataset penelitian.

4. KESIMPULAN

Penelitian ini membandingkan dua algoritma klasifikasi, yaitu *Decision Tree* dan *Random Forest* untuk menemukan algoritma yang paling optimal dalam mendeteksi serangan Bot DDos. Penelitian dilakukan dengan menggunakan dua dataset sekunder dan hasilnya menunjukan bahwa algoritma *Random Forest* unggul tipis secara hasil akurasi dibandingkan dengan algoritma *Decision Tree*, dan algoritma *Decision Tree* unggul jauh secara durasi waktu dibandingkan dengan algoritma *Random Forest*. Hasil tersebut terjadi pada kedua dataset dan tidak ada hasil yang bertolak-belakang antara kedua dataset tersebut.

Berdasarkan hasil penelitian ini, dapat disimpulkan bahwa algoritma *Decision Tree* menjadi algoritma yang direkomendasikan jika membutuhkan hasil yang cepat, hingga 15x lipat dibandingkan *Random Forest* pada kedua

dataset penelitian, dengan tetap menjaga tingkat akurasi yang cukup tinggi, dan algoritma *Random Forest* menjadi algoritma yang direkomendasikan jika membutuhkan hasil akurasi yang lebih tinggi dengan selisih berkisar 0.0001 dibandingkan *Decision Tree* pada kedua dataset. Penelitian ini diharapkan dapat menjadi sumber referensi untuk memilih algoritma yang tepat sesuai kebutuhan dari masing-masing pengguna.

Adapun keterbatasan dalam penelitian ini adalah dataset yang digunakan terbatas pada dataset dengan jumlah data di bawah 1 juta baris dikarenakan keterbatasan sumber daya untuk mengolah data dalam jumlah besar. Oleh karena itu, penelitian selanjutnya dapat menggunakan dataset dengan jumlah data di atas 1 juta baris untuk menguji tingkat akurasi serta waktu pemrosesan agar didapatkan hasil yang lebih akurat. Selain itu, penelitian selanjutnya dapat menambahkan algoritma lainnya guna memvalidasi keunggulan dari algoritma *Decision Tree* dan *Random Forest.*

DAFTAR PUSTAKA

[1] M. R. Aisy, "Tren Bisnis Online: Analisis Perubahan Konsumen Dan Strategi Pengembangan Bisnis di Era Digital," *J. Compr. Sci.*, vol. 3, pp. 750–755, 2024.

[2] M. Wida Rahmayani, N. Hernita, A. Gumilang, and W. Riyadi, "Pengaruh Digital Marketing Terhadap Peningkatan Volume Penjualan Hasil Industri Rumahan Desa Cibodas," *Coopetition J. Ilm. Manaj.*, vol. 14, no. 1, pp. 131–140, 2023, doi: 10.32670/coopetition.v14i1.1428.

[3] D. Syahroni and Nurjaya, "Optimasi Metode Naïve Bayes dengan Particle Swarm Optimization untuk Sistem Deteksi Serangan D-Dos," *J. Pendidik. dan Konseling*, vol. 4, 2022.

[4] R. TEKiN, O. YAMAN, and T. TUNCER, "Decision Tree Based Intrusion Detection Method in the Internet of Things," *Int. J. Innov. Eng. Appl.*, vol. 6, no. 1, pp. 17–23, Jun. 2022, doi: 10.46460/ijiea.970383.

[5] M. S. Rafsanjani, V. Suryani, and R. R. Pahlevi, "Deteksi Serangan Botnet Pada Jaringan Internet of Things Menggunakan Algoritma Random Forest (RF)," *e-Proceeding Eng.*, vol. 9, no. 1, pp. 1862–1871, 2022.

[6] Y. Yanti, T. Hidayat, N. Nurhanif, N. Safana, and P. N. P. Anggayoni, "Deteksi Serangan Distributed Deniel of Service Pada Jaringan Sensor Nirkabel Menggunakan Support Vector Machine," *G-Tech J. Teknol. Terap.*, vol. 8, no. 4, pp. 2687–2697, Oct. 2024, doi: 10.70609/gtech.v8i4.5428.

[7] D. P. Sari, Z. Halim, I. Irlon, B. Waseso, and S. Saromah, "Implementasi Machine Learning untuk Deteksi Intrusi pada Jaringan Komputer," *J. Minfo Polgan*, vol. 13, no. 2, pp. 1389–1394, Sep. 2024, doi: 10.33395/jmp.v13i2.14074.

[8] I. Maulana and Alamsyah, "Optimalisasi Deteksi Serangan DDoS Menggunakan Algoritma Random Forest, SVM, KNN dan MLP pada Jaringan Komputer," *Indones. J. Math. Nat. Sci.*, vol. 46, pp. 83–92, 2023.

[9] A. T. Zy, A. T. Sasongko, and A. Z. Kamalia, "Penerapan Naïve Bayes Classifier, Support Vector Machine, dan Decision Tree untuk Meningkatkan Deteksi Ancaman Keamanan Jaringan," *Media Online)*, vol. 4, no. 1, pp. 610–617, 2023, doi: 10.30865/klik.v4i1.1134.

[10] S. Rabbani and D. Diana, "Prediksi Kategori Serangan Siber dengan Algoritma Klasifikasi Random Forest Menggunakan Rapidminer," *SMATIKA J.*, vol. 13, no. 02, pp. 284–293, Dec. 2023, doi: 10.32664/smatika.v13i02.934.

[11] K. B. Dasari and N. Devarakonda, "Detection of DDoS Attacks Using Machine Learning Classification Algorithms," *Int. J. Comput. Netw. Inf. Secur.*, vol. 14, no. 6, pp. 89–97, Dec. 2022, doi: 10.5815/ijcnis.2022.06.07.

[12] R. N. Ramadhon, A. Ogi, A. P. Agung, R. Putra, S. S. Febrihartina, and U. Firdaus, "Implementasi Algoritma Decision Tree untuk Klasifikasi Pelanggan Aktif atau Tidak Aktif pada Data Bank," *Karimah Tauhid*, vol. 3, 2024.

[13] Z. M. J. Nafis, R. Nazilla, R. Nugraha, and S. Uyun, "Perbandingan Algoritma Decision Tree dan K-Nearest Neighbor untuk Klasifikasi Serangan Jaringan IoT," *Komputika J. Sist. Komput.*, vol. 13, no. 2, pp. 245–252, Oct. 2024, doi: 10.34010/komputika.v13i2.12609.

[14] J. J. Praba and R. Sridaran, "An SDN-based Decision Tree Detection (DTD) Model for Detecting DDoS Attacks in Cloud Environment," *IJACSA) Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 7, 2022, [Online]. Available: www.ijacsa.thesai.org

[15] Wahyuni and Pitrasacha Adytia, "Perbandingan Algoritma Machine Learning Dalam Mendeteksi Serangan DDOS," *TEMATIK*, vol. 9, no. 2, pp. 161–166, Nov. 2022, doi: 10.38204/tematik.v9i2.1070.

[16] D. P. Sinambela, H. Naparin, M. Zulfadhilah, and N. Hidayah, "Implementasi Algoritma Decision Tree dan

Random Forest dalam Prediksi Perdarahan Pascasalin," *J. Inf. dan Teknol.*, vol. 5, no. 3, pp. 58–64, Sep. 2023, doi: 10.60083/jidt.v5i3.393.

[17] E. A. Winanto, Y. Novianto, S. Sharipuddin, I. S. Wijaya, and P. A. Jusia, "Peningkatan Performa Deteksi Serangan Menggunakan Metode PCA dan Random Forest," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 11, no. 2, pp. 285–290, Apr. 2024, doi: 10.25126/jtiik.20241127678.

[18] M. Alduailij, Q. W. Khan, M. Tahir, M. Sardaraz, M. Alduailij, and F. Malik, "Machine-Learning-Based DDoS Attack Detection Using Mutual Information and Random Forest Feature Importance Method," *Symmetry (Basel).*, vol. 14, no. 6, Jun. 2022, doi: 10.3390/sym14061095.

[19] S. Chorev *et al.*, "Deepchecks: A Library for Testing and Validating Machine Learning Models and Data," *J. Mach. Learn. Res.*, vol. 23, pp. 1–6, 2022.

[20] J. Kaliappan, A. R. Bagepalli, S. Almal, R. Mishra, Y. C. Hu, and K. Srinivasan, "Impact of Cross-Validation on Machine Learning Models for Early Detection of Intrauterine Fetal Demise," *Diagnostics*, vol. 13, no. 10, pp. 1–22, 2023, doi: 10.3390/diagnostics13101692.