Perbandingan Kinerja Model Deteksi Serangan Pada Intrusion Detection System Dengan Tuning Hyperparameter

Dimas Shafir Alfirdausi Wahyu Purnomo Putra[1], I Made Suartana[2],

[1,2] Teknik Informatika/Teknik Informatika, Universitas Negeri Surabaya
[1]dimas_18062@mhs.unesa.ac.id
[2]imadesuartana@unesa.ac.id

***Abstrak*— Banyaknya penggunaan internet membuat banyak sektor mengalami kemudahan dan kemajuan, baik pada Pendidikan sampai pertahanan, akan tetapi dengan berkembangnya internet, makin berkembang pula cakupan kejahatan yang merambat pada piranti digital atau internet yang disebut *cyber crime*. Dengan bermunculannya kejahatan siber, maka muncul pula pertahan siber atau c*ybersecurity* dengan salah satu Upaya untuk mengatasi *cyber crime* adalah dengan menggunakan *intrusion detection system* atau sistem pendeteksian serangan. dengan berkembangnya teknologi, diberlakukan pula teknologi *machine learning* pada data serangan pada *intrusion detection system* guna mengetahui apakah ada serangan pada sistem atau tidak. Penggunaan *machine learning* pada *cybersecurity* akan membuka lembaran baru pada lini pertahanan siber yang cakupan perlindungannya menjadi semakin lebar dengan makin banyaknya model serangan. Dengan Random Forest dan Decision Tree yang merupakan *supervised learning* dan memiliki keandalan pada klasifikasi, dapat membantu proses pendeteksian serangan pada sistem dengan melakukan *training* dan *testing* pada dataset hasil *traffic*. Lebih jauh, dengan mengandalkan Random Forest dan Decision Tree dengan dioperasikan Hyperparameter Tuning, keandalan akan semakin meningkat. Dengan menggunakan algoritma *machine learning*, dataset dapat di *training* dengan akurasi 99% dengan akurasi prediksi atau *test* sebesar 86% pada decision tree hingga 87% pada random forest, terbukti handal untuk mendeteksi serangan pada dataset. Dengan perbandingan akurasi 1% pada prediksi, dapat dikatakan bahwa algoritma random forest lebih handal dalam pendeteksian serangan.**

***Kata Kunci*— IDS, Machine Learning, Random Forest, Decision Tree, Hyperparameter Tuning, Google Colab.**

I. PENDAHULUAN

5,473,055,736 orang di seluruh dunia menggunakan internet menurut *website* internetworldstats.com. Beradasarkan data dari sumber yang sama, sampai dengan juli 2022 pengguna internet di indonesia adalah 212,354,070 pengguna.Pengguna yang sekian banyak membuat internet menjadi kebutuhan bagi warga negara Indonesia dan dunia. Internet adalah teknologi modern yang berperan penting pada era globalisasi terutama pada dunia Pendidikan [8].Orang bisa mencari apapun dan menjadi siapapun yang dia mau, di berbagai situs atau platform, di berbagai perangkat baik pc ataupun *mobile*. Semua orang bisa menjelajahi dunia hanya dengan sentuhan tangan. Dengan kebebasan yang sangat mudah digapai tentu saja menghasilkan banyak hal positif seperti pendidikan yang maju, lapangan pekerjaan yang luas, dan wawasan yang lebih luas. Tidak hanya dampak positif saja, dampak negatif bisa ditimbulkan. Adanya kebebasan tentu membuka gerbang kebebasan pula bagi tindakan tindakan yang kurang baik dan tergolong kejahatan, atau biasa disebut kejahatan *cyber* atau *cybercrime*, meliputi pornografi anak, *cyberstalking*, pencurian identitas, pencurian kartu kredit, terorisme dunia maya, penjualan narkoba, kebocoran data, konten seksual vulgar, *phishing*, dan bentuk peretasan dunia maya lainnya [7]. Dengan serangkaian kejadian, maka perlu dilakukan studi untuk mempelajari serangan serangan *cyber* yang pernah terjadi.

Serangan siber atau *cyber attack* adalah serangan yang dilakukan seseorang atau beberapa orang yang melakukan serangkaian aktifitas untuk merusak sebuah sistem yang mengincar data yang bersifat rahasia. Serangan siber yang sudah diketahui contohnya dalah *hacking*, *phising*, dan perusakan situs web [7]. Berbagai cara sudah dilakukan untuk mengatasi dan mencegah serangan serangan cyber yang mungkin terjadi. Penanganan bisa dilakukan jika serangan itu sudah terdeteksi dan sudah terbukti melakukan serangan. Karena jika tidak terdeteksi, Korban akan mengalami kerugian mulai dari pencurian data, penghapusan data, penguncian data, hingga pemalsuan data. Juga ada serangan dimana penyerang akan menyusupi sebuah program atau *file* yang berisikan virus yang dapat menghancurkan data dalam perangkat korban apabila tidak dihapus segera. Akan sulit jika mengetahui serangan pada saat serangan itu diluncurkan, tetapi apapun

yang melewati jaringan internet akan terekam di *network traffic*. Dengan melakukan deteksi di *network traffic* maka serangan dapat diketahui. Dengan melakukan model training menggunakan algoritma machine learning kita bisa mengetahui kebiasaan yang dilakukan oleh serangan tertentu dan nantinya bisa diketahui itu adalah serangan atau bukan, selanjutnya dapat dilakukan penindakan pada serangan siber yang mengancam keamanan siber.

Cybersecurity atau keamanan siber adalah serangkaian tindakan untuk melindungi informasi dan data dari serangan dari sumber luar. *cybersecurity* meliputi perlindungan keamanan jaringan, sever, intranet, dan perangkat komputer [1]. Keamanan jaringan adalah salah satu poin dalam *cybersecurity* atau keamanan siber yang merupakan suatu atau serangkaian kegiatan yang berisikan instruksi dan instrumen yang berkenaan dengan keamanan diantaranya adalah deteksi. deteksi adalah salah satu kegiatan paling awal dari serangkaian proses keamanan jaringan untuk mendeteksi dan mengetahui adakah suatu proses, upaya, ataupun kegiatan yang mencurigakan pada jaringan. Salah satu bentuk aktfitasnya ialah dengan melakukan upaya masuk pada jaringan secara berulang, atau memberikan request yang berulang dan banyak pada suatu jaringan atau server sehingga mengakibatkan gangguan ataupun kerusakan pada perangkat jaringan. Untuk meminimalisir hal hal yang demikian, haruslah menggunakan tata cara dan instrumen yang tepat untuk deteksi, salah satunya menggunakan *wireshark* yang merupakan aplikasi yang dulunya bernama *ethereal*, digunakan untuk troubleshooting dan menganalisa jaringan computer [2]. berfungsi untuk mengawasi dan merekam lalu lintas data lewat jaringan internet dalam bentuk *log* proses yang berisikan informasi dari berbagai lapisan *OSI Layer* yang bisa digunakan sebagai tolok ukur untuk mengetahui apakah ada hal yang mencurigakan terjadi disana atau tidak. Untuk mempermudah proses deteksi, sudah ada beberapa metode untuk melakukan scanning dan pemeriksaan terhadap *log* atas jaringan, salah satunya adalah dengan menggunakan algoritma *machine learning*.

Referensi [4] mengatakan *machine learning* berkembang di tahun tahun ini pada bidang data analis dan komputasi yang memungkinkan aplikasi dapat berfungsi sesuai kepintaran yang dibangun untuk sistem dan dapat berkembang. *Machine learning* memiliki banyak jenis, salah satunya adalah *supervised learning* yang meliputi klasifikasi dan regresi, dengan data yang sudah memiliki label sebagai bahan utama. Label memiliki peran penting dalam hal pendeteksian karena menjadi salah satu faktor penentu sebuah *network traffic* disebut serangan, karena itu klasifikasi menjadi metode yang dipilih, terutama dengan algoritma Random Forest dan Decision tree. Random Forest dan Decision Tree merupakan dua dari beberapa algoritma klasifikasi yang memiliki performa yang terbaik menurut beberapa penelitian tentang pendeteksian serangan. Pemilihan algoritma *machine learning* untuk model pendeteksian adalah hal penting, disamping itu penerapan proses pembuatan model juga penting, *preprocessing* data adalah salah satunya, proses itu merupakan sebuah awal dari proses pembuatan model atau kata lainnya adalah persiapan. *Preprocessing* bertujuan untuk menyiapkan data agar lebih matang untuk nantinya bisa dioperasikan menjadi model deteksi serangan. Selain itu ada beberapa hal yang dilakukan seperti *data split* atau pembagian data dan tuning hyperparameter atau pemilihan hyperparameter, keduanya dilakukan untuk membuat model deteksi yang lebih handal.

Referensi [5] Melakukan penelitian tentang penerapan algoritma machine learning random forest dan dalam penelitian tersebut didapati bahwasannya algoritma random forest memiliki akurasi 99% pendeteksian terhadap serangan yang ada pada dataset yang dibahas. Penelitian lain yang serupa yaitu dengan melakukan optimasi data dan algoritma random forest untuk pendeteksian serangan dengan hasil performa yang terbaik dalam mendeteksi anomali seperti *DoS*, *Analysis, Backdoor*, dan *Worm* [16]. Dengan berbagai penelitian yang sudah dilakukan, bisa dikatakan bahwa topik tentang keamanan jaringan memang sangat penting untuk kelangsungan hidup berinternet sekarang ini. Dengan hasil hasil yang sudah dilakukan oleh peneliti terdahulu, saya mendapati bahwa algoritma *machine learning* yang digunakan untuk IDS mendapati hasil yang baik. Dengan adanya penelitian terdahulu, akan menjadi fondasi untuk penelitian kali ini yang juga menggunakan topik deteksi pada jaringan yang digunakan untuk menjaga keamanan jaringan.

Tujuan penelitian yang akan dilakukan adalah untuk menggunakan algoritma machine learning untuk kepentingan pendeteksian serangan yang tejadi pada jaringan, dengan nantinya akan dilakukan klasifikasi akan serangan serangan yang ada, dan dengan menggunakan beberapa metode guna untuk perbandingan performa, agar nantinya dapat didapati hasil bagaimana cara deteksi serangan pada *log* jaringan dengan menggunakan Model dan mengetahui model apakah yang memiliki keandalan dibanding model yang lain.

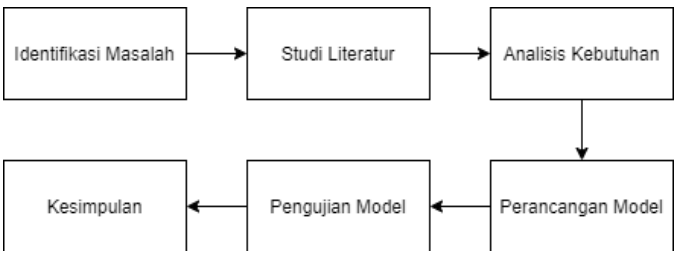
II. METODE PENELITIAN

A. Jenis Penelitian

Metode yang diterapkan pada penelitian ini adalah metode experimental design. Penerapan metode bertujuan untuk menganalisa kinerja model machine learning guna mendeteksi serangan pada dataset sdn network.

B. Rancangan Penelitian

Penelitian ini akan mengalokasikan tahapan yang jelas untuk mencapai tujuan tersebut. Tahap pertama adalah identifikasi masalah dimana akan dilakukan menentukan masala hapa yang menjadi dasar penelitian yaitu mengenai pendeteksian cyber attack dengan menggunakan machine learning. Tahap kedua yaitu studi literatur dengan melakukan pencarian terhadap sumber literatur yang berhubungan dengan penelitian yang digunakan sebagai pendukung dann juga acuan penelitian, guna penelitian bisa berjalan dengan baik. Ketiga analisis kebutuhan, menentukan kebutuhan yang dipelukan untuk melaksanakan penelitian, direncakan dan dipikirkan secara menyeluruh agar tujuan dari penelitian dapat tercapai. Selanjutnya adalah Perancangan Model, di mana akan dilakukan perancangan model yang bertujuan untuk menggunakan algoritma machine learning untuk model pendeteksian serangan pada dataset agar nantinya ketika ada dataset yang dicurigai mengandung serangan serangan siber dapat diketahui dan diatasi dengan cara yang efisien. Setelah model sudah dibuat maka akan dilakukan pengujian model dimana model akan diuji dengan dataset yang sudah ditentukan. Parameter pengujian yang akan dilakukan adalah menguji dataset dan dioperasikan menggunakan model yang sudah dibuat dengan algoritma yang sudah diimplementasikan pada model dan dapat mendeteksi serangan apa saja yang ada pada dataset, kemudian akan diujikan berapa persen akurasi deteksi algoritma yang digunakan. Terakhir akan dilakukan Kesimpulan, Setelah dilakukan serangkaian perencanaan, implementasi, pengujian pada model, maka akan didapati kesimpulan dan pemberian saran dari serangkaian penelitian yang telah dilakukan berdasarkan rumusan masalah dan batasan, hasil dari pengujian sistem dapat menjadi jawaban dari masalah yang dijelaskan serta dataset dari penelitian, yang diharapkan nantinya dapat berguna dalam model deteksi serangan ada dataset dan dapat menjadi referensi penelitan yang selanjutnya.



Gbr 1. Alur Metode Penelitian

C. Analisis Kebutuhan

Analisis kebutuhan meripakan analisis yang dibutuhkan untuk menentukan detail kebutuhan pada penelitian pendeteksian serangan dengan menggunakan algoritma machine learning. Penelitian ini menggunakan algoritma machine learning yang dijalankan pada aplikasi google yaitu google collab yang nantinya digunakan juga untuk menguji dataset. Sehingga dibutuhkan sebuah perangkat yang dapat mendukung agar penelitian ini dapat berjalan sesuai tujuan, berikut merupakan kebutuhan yang dibagi menjadi beberapa bagian, yaitu:

1) Spesifikasi Perangkat Keras (Hardware)

Perangkat keras yang digunakan dalam penelitian guna mewujudkan tujuan penelitian yaitu perangkat laptop sebagai uji coba dengan spesifikasi berikut :

Tabel 1. Spesifikasi Perangkat Keras

No	Perangkat Keras	Keterangan
1	Processor	Intel Core i7-8700H
2	Memory Ram	12.00 GB
3	Storage	SSD 128 GB
4	Sistem Operasi	Windows 10 Enterprise 64-bit

2) Spesifikasi Perangkat Lunak (Software)

Perangkat lunak berfungsi untuk pengoperasian sistem pada penelitian ini. Pada penelitian ini digunakan adalah :

Tabel 2. Spesifikasi Perangkat Lunak

No	Perangkat Lunak	Keterangan
1	Sistem operasi	Windows 10 Enterprise 64-bit
2	Aplikasi	Google Chrome versi 107.0.5304.88 (Official Build) (64-bit)
3	Website	Google Colab

D. Perancangan Model

Perancangan Model dilakukan untuk membuat desain perencaan sistem yang akan digunakan untuk menguji dataset agar mencapai tujuan penelitian yang sudah direncanakan. Berikut adalah pembahasannya :

1) Penentuan Dataset

Sebelum membangun Model yang akan diteliti, sebelumnya akan ditetukan Dataset yang akan dioperasikan Algoritma Machine learning, yang dalam penelitian ini adalah:

Tabel 3. Spesifikasi Dataset

No	Nama	Keterangan
1	Dataset	UNSW-NB15
2	Jenis Dataset	Dataset Berlabel
3	Pembuat	IXIA PerfectStorm tool di Cyber Range Lab of the Australian Centre for Cyber Security (ACCS)
4	Tahun	2015

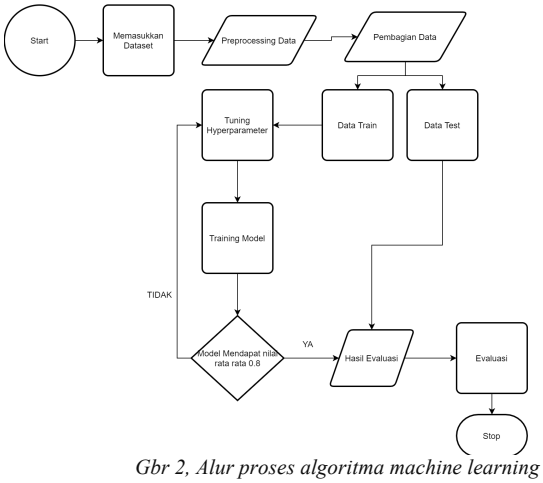
2) Penentuan Algoritma Machine Learning:

Setelah penentuan dataset, selanjutnya akan ditentukan algoritma machine learning yang akan dioperasikan menjadi model, dalam penelitian ini yang digunakan adalah:

Tabel 5. Algoritma Machine Learning

No	Nama	Keterangan
1	Random forest	Supervised Learning
2	Decision Tree	Supervised Learning

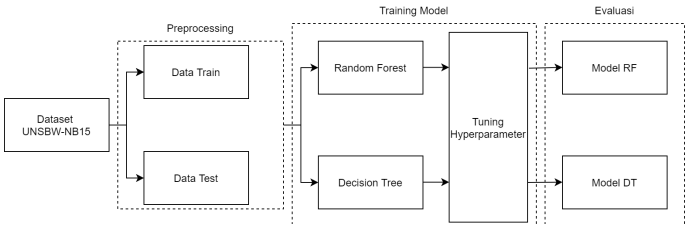
Dengan algoritma yang sudah dipilih, maka akan dijabarkan pula proses yang akan dilakukan yaitu memasukkan dataset lalu akan dilakukan preprocessing data yang akan menjadi data test dan data train selanjutnya akan dilakukan training dengan menggunakan hyperparameter, setelah hasil didapatkan, maka akan dilakukan evaluasi.



Gbr 2, Alur proses algoritma machine learning

3) Skema Perancangan Model:

Model yang akan dibangun menggunakan bahasa python dengan menggunakan platform jupyter yang disediakan oleh google colab. Seperti pada gambar 2, Dataset yang digunakan adalah UNSW-NB15 yang sudah dilakukan preprocessing data berupa data cleaning dari kolom yang tidak diperlukan untuk model training dengan menghapus secara manual kolom yang korelasinya rendah atau tidak saling berkorelasi, selanjutnya diberlakukan Standard scaler. data train akan ditraining dengan algoritma Random Forest dan decision tree. Tuning hyperparameter yang akan dilakukan pada Random Forest adalah pada n_estimator dan criterion, dan pada Decision Tree adalah criterion dan splitter. akan menghasilkan model deteksi serangan, dan akan di test menggunakan data test, hasilnya akan berupa prosentase untuk akurasi, precision, recall, dan f-1 score. Penilaian akhir yang diharapkan sama atau melebihi dari penelitian yang sudah pernah dilakukan dengan menggunakan algoritma random forest yaitu 0.86 (Ren et al., 2019) , 0.99 (Park et al., 2018), dan Decision Tree yaitu 0.85 (Ren et al., 2019), 0.99 (Khraisat et al., 2018).



Gbr 3. Skema Perancangan Model

III. *HASIL DAN PEMBAHASAN*

Tahap ini membahas mengenai hasil Perbandingan Kinerja Model Deteksi Serangan pada Intrusion Detection System dengan Tuning Hyperparameter. Hasil penelitian dan pembahasan akan menjelaskan hasil implementasi sistem yang telah dirancang berdasarkan permasalahan dan tujuan dari penelitian yang telah dirumuskan dengan proses implementasi

yaitu mulai dengan pembuatan notebook di Google Colab dan pengujian model hingga menemukan hasil.

A. Implementasi

1) Inisiasi dan Menyambungkan Dataset di Notebook Google Colab

Setelah Notebook sudah siap untuk digunakan, maka selanjutnya akan dilakukan koding menggunakan Bahasa python yang diawali dengan inisiasi modul apa saja yang akan digunakan.

Inisiasi

```
[ ] import zipfile,os
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import time
    import string

    from sklearn.utils.class_weight import compute_class_weight
```

Gbr 4. Inisiasi Modul

Beberapa modul yang digunakan antara lain adalah zipfile,os , pandas, numpy, matplotlib.pyplot, seaborn, time, string, dan compute_class_weight. Selanjutnya akan dilakukan penyambungan dengan dataset yang akan diujikan.

Persiapan Dataset

```
[ ] #download data
    from google.colab import drive
    drive.mount("/content/drive/", force_remount=True)
```

```
Mounted at /content/drive/
```

Gbr 5, Persiapan Dataset

Gbr 5 adalah kode untuk melakukan koneksi dengan drive atau google drive, karena dataset yang akan diujikan saya tempatkan pada google drive guna memudahkan dalam proses penelitian.

```
[ ] csv="/content/drive/MyDrive/DATASET/archive/UNSW_NB15_testing-set.csv"
    d_train = pd.read_csv(csv)
    d_train.head()
```

Gbr 6, Pengeluaran Data Train

```
[ ] csv="/content/drive/MyDrive/DATASET/archive/UNSW_NB15_training-set.csv"
    d_test = pd.read_csv(csv)
    d_test.head()
```

Gbr 7, Pengeluaran Data Test

Setelah dataset tersambung, maka selanjutnya akan dilakukan pengeluaran data train (gbr 6), dan data test (gbr 7) yang nantinya akan digunakan untuk pengoperasian model machine learning.

2) Preprocessing Data

Untuk melanjutkan penelitian dengan mengoperasikan model, maka dataset perlu diproses awal atau preprocessing dengan normalisasi yaitu untuk memproses data yang berada di antara nilai 0 dan 1 seperti pada gambar dibawah.

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Gbr 8, Normalisasi data

B. Pengujian

1) Model 1 Random Forest

Setelah dataset kita olah awal, maka dataset siap diopersikan dengan model, sesuai dengan penelitian, model yang akan digunakan adalah Random Forest dan Decision Tree.

```
[ ] from sklearn.ensemble import RandomForestClassifier
```

```
[ ] train_accuracy=[]
    test_accuracy=[]
    train_time=[]
    test_time=[]
    names = ['Random Forest','Decision Tree']
```

```
[ ] model1 = RandomForestClassifier()
```

Gbr 9. Tampilan Broadcast Katalog

Gbr 9 merupakan proses untuk inisiasi Random Forest dengan import Randomforestclassifier dari sklearn. Lalu menegaskan bahwa model 1 adalah menggunakan randomforestclassifier.

```
[ ] start_time = time.time()
    model1.fit(X_train, Y_train.values.ravel())
    end_time = time.time()
```

```
[ ] train_time.append(end_time-start_time)
    print("Training time: ",train_time[0])
```

```
Training time:  46.17248725891113
```

Gbr 10, Kode Training Random Forest

Setelah inisiasi, maka yang akan kita operasikan pertama adalah dengan melakukan proses training dengan dataset train yang sudah dibagi menjadi x dan y, atau dalam penelitian ini adalah x_train dan y_train. Berdasarkan gambar 10, hasil waktu training adalah 46.17s.

```
[ ]  start_time = time.time()
     Y_test_pred1 = model1.predict(X_test)
     end_time = time.time()
```

```
     test_time.append(end_time-start_time)
     print("Testing time: ",test_time[0])
```

```
Testing time:  1.062246322631836
```

Gbr 11, Kode Test Random Forest

Setelah model melakukan train dengan dataset train, selanjutnya pada gbr 11 dilakukan proses testing dengan mengoperasikan hasil training dengan dataset test atau x_test. Dengan waktu test 1.06s.

```
[ ]  train_accuracy.append(model1.score(X_train, Y_train))
     test_accuracy.append(model1.score(X_test,Y_test))
     print("Train score is:", train_accuracy[0])
     print("Train Accuracy: "+ "{:.2%}".format(train_accuracy[0]))
     print("Test score is:",test_accuracy[0])
     print("Test Accuracy: "+ "{:.2%}".format(test_accuracy[0]))
```

```
Train score is: 0.9981578752259882
Train Accuracy: 99.82%
Test score is: 0.8715809162901423
Test Accuracy: 87.16%
```

Gbr 12, Hasil Train dan Test Random Forest

Training dan testing yang sudah dilakukan tentu saja akan diidentifikasi hasil dari training dan testing, dengan hasil yaitu score dan accuracy. seperti pada gbr 12 dapat diketahui bahwa hasil akurasi train yaitu 99.82% dan hasil akurasi test yaitu 87.16%.

2) Model 2 Decision Tree

Setelah model 1 yaitu Random Forest, maka dataset akan dioperasikan lagi dengan model 2 yaitu Decision Tree. Yang akan diawali dengan inisiasi.

```
[ ]  #Decision Tree
     from sklearn.tree import DecisionTreeClassifier
```

```
[ ]  model2 = DecisionTreeClassifier()
```

```
criterion="entropy", max_depth = 4,class_weight='balanced'
```

Gbr 13. Inisiasi model Decision Tree

Gbr 13 adalah proses untuk inisiasi Decision Tree dengan import DecisionTreeClassifier dari sklearn. Lalu menegaskan bahwa model 2 adalah menggunakan DecisionTreeClassifier.

```
[ ]  start_time = time.time()
     model2.fit(X_train, Y_train.values.ravel())
     end_time = time.time()
```

```
[ ]  train_time.append(end_time-start_time)
     print("Training time: ",train_time[1])
```

```
Training time:  4.311499118804932
```

Gbr 14. Kode Training Decision Tree

Setelah inisiasi, maka yang akan kita operasikan pertama adalah dengan melakukan proses training dengan dataset train yang sudah dibagi menjadi x dan y, atau dalam penelitian ini adalah x_train dan y_train. Pada gbr 14, diketahui waktu train adalah 4.31s.

```
[ ]  start_time = time.time()
     Y_test_pred2 = model2.predict(X_test)
     end_time = time.time()
```

```
[ ]  test_time.append(end_time-start_time)
     print("Testing time: ",test_time[1])
```

```
Testing time:  0.032270193099975586
```

Gbr 15, Kode Test Decision Tree

Setelah model melakukan train dengan dataset train, selanjutnya pada gambar 4.18 dilakukan proses testing dengan mengoperasikan hasil training dengan dataset test atau x_test.

```
train_accuracy.append(model2.score(X_train, Y_train))
test_accuracy.append(model2.score(X_test,Y_test))
print("Train score is:", train_accuracy[1])
print("Train Accuracy: "+ "{:.2%}".format(train_accuracy[1]))
print("Test score is:",test_accuracy[1])
print("Test Accuracy: "+ "{:.2%}".format(test_accuracy[1]))
```

```
Train score is: 0.9981635783986632
Train Accuracy: 99.82%
Test score is: 0.8603945003157946
Test Accuracy: 86.04%
```

Gbr 16, Hasil train dan test Decision Tree

Training dan testing yang sudah dilakukan tentu saja akan diidentifikasi hasil dari training dan testing, dengan hasil yaitu score dan accuracy. seperti pada gbr 16 dapat diketahui bahwa hasil akurasi train adalah 99.82% dan akurasi test adalah 86.04%

3) Perbandingan Model

Setelah Kedua model sudah dioperasikan masing masing, sebelum dilakukan operasi selanjutnya, akan dilakukan pemeriksaan terhadap hasil train maupun test dari masing masing model yang kemudian akan dibandingkan.

Tabel 6. Perbandingan Model

NO.	Model	Training Time	Training Accuracy	Test Time	Test Accuracy
1.	Random Forest	46.17 s	99.82%	1.06 s	87.16%
2.	Decision Tree	4.31 s	99.82%	0.03 s	86.04%

Berdasarkan Tabel 6, kedua model sudah dioperasikan masing masing, namun model Random Forest memiliki waktu train dan test yang lebih tinggi dibandingan dengan decision tree, tetapi sama pada akurasi training pada 99.82% dengan angka akurasi test yang berbeda dengan Random Forest pada 87.16% sementara Decision Tree pada 86.04%.

4) Hyperparameter Tuning Random Forest

Pada gbr 11 sudah dilakukan pengoperasian test model dengan menggunakan Random Forest dengan parameter default yang disediakan oleh SKlearn dengan hasil yang sudah digambarkan pada gbr 12. setelah mendapatkan hasil test tersebut, selanjutnya adalah mengoperasikan Random Forest dengan Hyperparameter Tuning seperti pada gbr 17 dibawah ini

```
from sklearn.model_selection import GridSearchCV, train_test_split, RepeatedStratifiedKFold, cross

cv_method = RepeatedStratifiedKFold(n_splits=5, n_repeats=10, random_state=42)

param_grid_rf = {'n_estimators':np.arange(50,150,10),'criterion':['gini','entropy']}
model1_hpt = GridSearchCV(RandomForestClassifier(), param_grid=param_grid_rf, scoring='accuracy')
start = time.time()
model1_hpt.fit(X_train, Y_train.values.ravel())
end_train = time.time()
y_pred1_hpt = model1_hpt.predict(X_test)
end_predict = time.time()

accuracy_hpt=[]
# name = ['Random Forest','Decision Tree','HPT_RF','HPT_DT']
```

Gbr 17. Kode Hyperparameter Tuning Random Forest

Di gbr 17 dapat diperhatikan terdapat hal yang berbeda seperti pada Gbr 7 atau iniasi Random Forest yaitu adalah diberlakukan Hyperparameter Tuning pada n_estimator dan criterion. Yang terjadi adalah operasi model akan dilaksanakan sesuai yang sudah diterangkan pada n_estimator dan criterion yang sebelumnya menggunakan default dari SKlearn.

```
accuracy_hpt.append(accuracy_score(Y_test, y_pred1_hpt))
recall = recall_score(Y_test, y_pred1_hpt, average='weighted')
precision = precision_score(Y_test, y_pred1_hpt, average='weighted')
f1s = f1_score(Y_test, y_pred1_hpt, average='weighted')
print("Best Parameters:",model1_hpt.best_params_)
print("Accuracy: "+ "{:.2%}".format(accuracy_hpt[0]))
print("Recall: "+ "{:.2%}".format(recall))
print("Precision: "+ "{:.2%}".format(precision))
print("F1-Score: "+ "{:.2%}".format(f1s))
print("time to train: "+ "{:.2f}".format(end_train-start)+" s")
print("time to predict: "+"{:.2f}".format(end_predict-end_train)+" s")
print("total: "+"{:.2f}".format(end_predict-start)+" s")
model_performance.loc['RF_HPT'] = [accuracy_hpt[0], recall, precision, f1s,end_trai
```

```
Best Parameters: {'criterion': 'entropy', 'n_estimators': 80}
Accuracy: 87.20%
Recall: 87.20%
Precision: 88.89%
F1-Score: 86.89%
time to train: 3616.42 s
time to predict: 0.79 s
total: 3617.21 s
```

Gbr 18 Hasil Hyperparameter Tuning Random Forest

Setelah dilakukan Hyperparameter Tuning dan model dijalankan, selain perbandingan parameter, perbandingan yang selanjutnya adalah waktu operasi model yang awalnya 46.17 s pada training dan 1.06 s pada test (gambar 4.14) menjadi 3613.42 s pada training dan 0.79 s pada test, dapat dilihat perbandingan yang signifikan pada training dimana pada Hyperparameter tuning lebih lambat yaitu sebanyak 3567.25 s dikarenakan parameter yang sudah disesuaikan dan model harus menjalankan sesuai apa yang diatur dalam hyperparameter tuning, maka dari itu proses training menjadi sangat lama. Sedangkan pada test, perbandingan yang terjadi tidak terlalu signifikan tetapi test pada Hyperparameter Tuning menjadi lebih cepat 0.27 s dibandingkan model dengan parameter default.

Selain waktu operasi model, hasil akurasi model default dan hyperparameter ada sedikit perbedaan dimana hyperparameter adalah 87.20% dan hasil model default adalah 87.16% dengan akurasi yang berbeda menjadikan model dengan hyperparameter tuning lebih baik dengan perbedaan sebesar 0.04% dibanding hasil model default random forest. Yang terakhir hasil yang ditampilkan adalah parameter mana yang terbaik untuk dioperasikan pada Random forest, yaitu criterion : entropy dan n_estimators: 80 .

5) Hyperparameter tuning Decision Tree

Pada gbr 15 sudah dilakukan pengoperasian test model dengan menggunakan Decision Tree dengan parameter default yang disediakan oleh SKlearn dengan hasil yang sudah digambarkan pada Gbr 16. setelah mendapatkan hasil test tersebut, selanjutnya adalah mengoperasikan Decision Tree dengan Hyperparameter Tuning seperti pada gambar dibawah ini.

```
param_grid_dt = {'criterion':['gini','entropy'], 'splitter':['best','ra

model_2 = GridSearchCV(DecisionTreeClassifier(), param_grid=param_grid_
model_2.fit(X_train, Y_train.values.ravel())
start = time.time()
end_train = time.time()
y_pred2_hpt = model_2.predict(X_test)
end_predict = time.time()
```

Gbr 19. Kode Hyperparameter Tuning Decision Tree

Di gbr 19 dapat diperhatikan terdapat hal yang berbeda seperti pada Gambar 14 atau iniasi Decision Tree yaitu adalah diberlakukan Hyperparameter Tuning pada criterion dan spliter. Yang terjadi adalah operasi model akan dilaksanakan sesuai yang sudah diterangkan pada criterion dan splitter yang sebelumnya menggunakan default dari SKlearn.

```
accuracy_hpt.append(accuracy_score(Y_test, y_pred2_hpt))
recall = recall_score(Y_test, y_pred2_hpt, average='weighted')
precision = precision_score(Y_test, y_pred2_hpt, average='weighted')
f1s = f1_score(Y_test, y_pred2_hpt, average='weighted')
print("Best Parameters:",model_2.best_params_)
print("Accuracy: "+ "{:.2%}".format(accuracy_hpt[1]))
print("Recall: "+ "{:.2%}".format(recall))
print("Precision: "+ "{:.2%}".format(precision))
print("F1-Score: "+ "{:.2%}".format(f1s))
print("time to train: "+ "{:.2f}".format(end_train-start)+" s")
print("time to predict: "+"{:.2f}".format(end_predict-end_train)+" s")
print("total: "+"{:.2f}".format(end_predict-start)+" s")
model_performance.loc['DT_HPT'] = [accuracy_hpt[1], recall, precision, f1s,
```

```
Best Parameters: {'criterion': 'gini', 'max_depth': 27, 'splitter': 'best'}
Accuracy: 86.69%
Recall: 86.69%
Precision: 87.62%
F1-Score: 86.47%
time to train: 0.00 s
time to predict: 0.02 s
total: 0.02 s
```

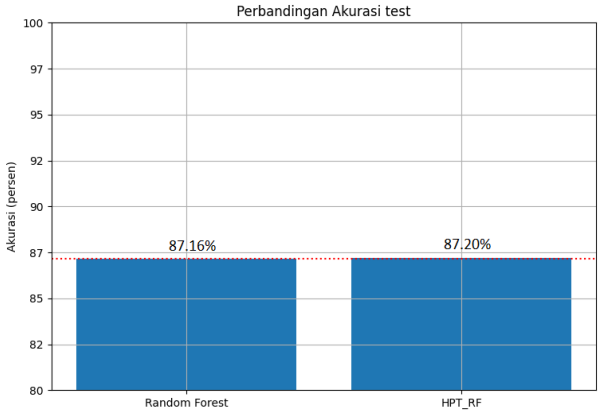
Gbr 20 Hasil Hyperparameter Tuning Decision Tree

Setelah dilakukan Hyperparameter Tuning dan model dijalankan, selain perbandingan parameter, perbandingan yang selanjutnya adalah waktu operasi model yang awalnya 4.3 s pada training dan 0.03 s pada test (gbr 15) menjadi 0.0 s pada training dan 0.02 s pada test, dapat dilihat perbandingan yang signifikan pada training dimana pada Hyperparameter tuning lebih cepat yaitu sebanyak 4.3 s dikarenakan parameter yang sudah disesuaikan dan model harus menjalankan sesuai apa yang diatur dalam hyperparameter tuning. Sedangkan pada test, perbandingan yang terjadi tidak terlalu signifikan tetapi test pada Hyperparameter Tuning menjadi lebih cepat 0.01 s dibandingkan model dengan parameter default.

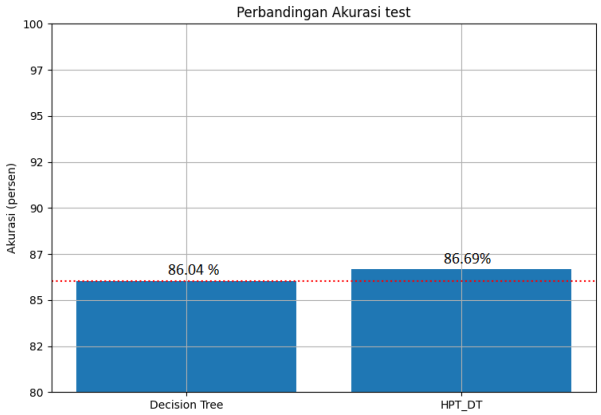
Selain waktu operasi model, hasil akurasi model default dan hyperparameter ada sedikit perbedaan dimana hyperparameter adalah 86.69% dan hasil model default adalah 86.04% dengan akurasi yang berbeda menjadikan model dengan hyperparameter tuning lebih baik dengan perbedaan sebesar 0.65%. Yang terakhir hasil yang ditampilkan adalah parameter mana yang terbaik untuk dioperasikan pada Decision Tree, yaitu Criterion: gini, max_depth: 27, dan splitter: best.

6) *Perbandingan Kinerja Model*

Setelah model random forest dan decision tree dioperasikan, juga hyperparameter dari masing masing model dijalankan pada test. Maka selanjutnya kan dilakukan perbandingan antara model tanpa hyperparameter dan model yang sudah dijalankan dengan hyperparameter.



Gbr 21. Perbandingan Model Random Forest dan Hyperparameter Random Forest



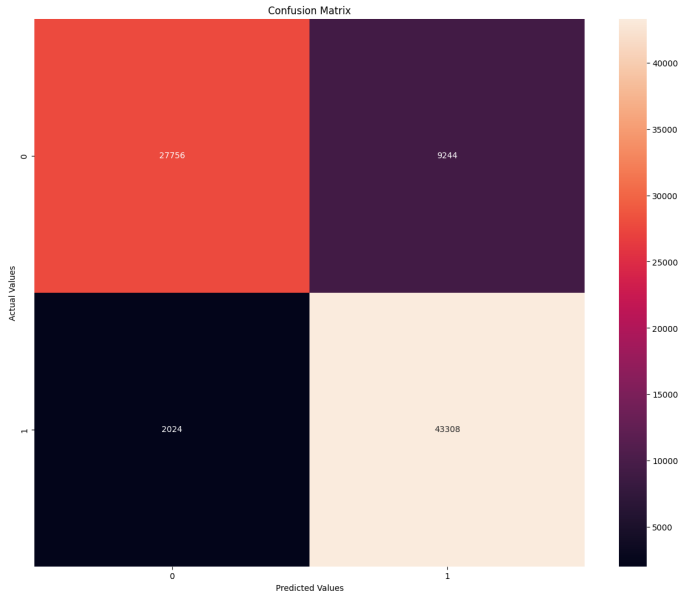
Gbr 22. Perbandingan Model Decision Tree dan Hyperparameter Decision Tree

Gbr 21 adalah perbandingan hasil pengoperasian dengan Akurasi Random Forest adalah 87.16% dan Akurasi Random Forest dengan Tuning hyperparameter adalah 87.20% dengan selisih 0.04% . pada gbr 22, Akurasi Decision Tree adalah 86.04%, dan Akurasi Decision Tree dengan Tuning Hyperparameter adalah 86.69% dengan selisih 0.65%.

7) *Confusion Matrix*

Rentetan operasi demi operasi sudah dilakukan dan hasil sudah ditampilkan pula pada gbr 21, namun hasil dari operasi yang membuktikan apakah hasil itu dapat dipercaya atau tidak masih belum ditampilkan, maka selanjutnya yang akan ditampilkan adalah hasil training dan test, apakah memang hasil tersebut membuktikan kinerja yang

digambarkan pada akurasi ataupun tidak. Berikut adalah gambarnya



Gbr 23. Confusion Matrix

Dapat dilihat pada gambar diatas, ada 2 kolom dan 2 baris dengan masing masing bertuliskan angka 0 pada baris pertama dan 1 pada baris kedua, serta 0 pada kolom pertama dan 1 pada kolom kedua. Dengan keterangan di sebelah kiri bertuliskan actual value atau dapat diartikan value yang sebenarnya, dan pada bagian bawah terdapat keterangan predicted value atau hasil yang diprediksikan. Dapat dilihat pada kolom pertama baris pertama terdapat angka 27664. Sebelumnya perlu diketahui bahwa kolom tersebut adalah pertemuan angka 0 dan 0 yang berarti kolom tersebut adalah untuk value 0 yang diprediksikan memiliki value 0 atau false. Dan metode tersebut membuktikan ada 27664 data yang value 0.

Selanjutnya pada kolom kedua baris pertama ada angka 9336 yang kolom tersebut adalah pertamuan antara value 0 pada actual value dan value 1 pada predicted value, pada model tersebut menyebutkan ada 9336 data value 0 yang diprediksikan memiliki value 1. Dilanjutkan pada kolom pertama baris kedua yang adalah pertemuan antara value 1 pada actual value dan value 0 pada predicted value menampilkan angka 2158, yang berarti model tersebut memprediksi ada 2158 data dengan value 0 padahal memiliki value 1. Yang terakhir pada kolom kedua baris kedua yang merupakan pertemuan value 1 pada actual value dan value 1 pada predicted value menampilkan angka 43174 yang berarti model tersebut memprediksi sebanyak 43174 data yang memiliki value 1 dengan value 1.

IV. KESIMPULAN

Berdasarkan penelitian Perbandingan Kinerja Model Deteksi Serangan pada Intrusion Detection System dengan Tuning Hyperparameter yang telah berhasil dilakukan, mendapatkan kesimpulan bahwa Algoritma machine learning Random Forest dan Decision Tree terbukti dapat diterapkan dalam Intrusion Detection System atau sistem deteksi serangan pada sebuah data dengan total data 175341 pada data train dan 82332 pada data test, didapati hasil 99.82% akurasi train pada random forest dan decision tree dengan waktu train 46.17s pada random forest dan 4.31s pada decision tree. Pada test masing masing random forest mendapati hasil 87.17% dan decision tree dengan 86.04%, dengan waktu masing masing 1.06s untuk random forest dan 0.03s pada decision tree. dengan selisih akurasi 0.51% didapati model tuning hyperparameter random forest lebih unggul dibanding model tuning hyperparameter decision tree. Tuning Hyperparameter yang telah dioperasikan pada random forest pada n_estimator dan criterion, serta pada decision tree pada criterion, splitter dan max_depth membuktikan memberikan perbedaan pada waktu train pada random forest yang sebelumnya memerlukan waktu train 46.17s menjadi 3616.42s dan waktu test yang sebelumnya 1.06s menjadi 0.79s , serta pada akurasinya yang sebelumnya 87.16% menjadi 87.20%. pada decision tree yang waktu train sebelum diberlakukan tuning hyperparameter adalah 4.31s menjadi 0.00s, dan test dengan waktu yang sebelumnya 0.03s menjadi 0.02s, serta akurasi yang sebelumnya 86.04% menjadi 86.69%. Maka dapat disimpulkan, Dengan Tuning Hyperparameter, waktu train menjadi lebih tinggi pada Random forest dan lebih rendah pada decision tree, dengan waktu test yang keduanya menjadi lebih singkat, dan dengan tuning hyperparameter keduanya terbukti memiliki akurasi yang menjadi lebih tinggi.

REFERENSI

[1] Khiralla, F. A. M. (2020). Statistics of cybercrime from 2016 to the first half of 2020. Int. J. Comput. Sci. Netw., 9(5), 252-261.
[2] Iqbal, H., & Naaz, S. (2019). Wireshark as a tool for detection of various LAN attacks. International Journal of Computer Science and Engineering, 7(05), 833-837.
[3] Zhou, Z. H. (2021). Machine learning. Springer Nature.
[4] Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. SN Computer Science, 2(3), 1-21.
[5] Park, K., Song, Y., & Cheong, Y. G. (2018, March). Classification of attack types for intrusion detection systems using a machine learning algorithm. In 2018 IEEE fourth international conference on big data computing service and applications (BigDataService) (pp. 282-286). IEEE.

[6] Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. applied sciences, 9(20), 4396.
[7] Al-Khater, W. A., Al-Maadeed, S., Ahmed, A. A., Sadiq, A. S., & Khan, M. K. (2020). Comprehensive review of cybercrime detection techniques. IEEE Access, 8, 137293-137311.
[8] Khamis, S., Ahmad, A., & Ahmad, M. (2019). A descriptive analytic model of internet usage and student performance. Technol. Manag, 4, 1-10.
[9] www.internetworldstats.com, diakses pada November 2022
[10] Aldweesh, A., Derhab, A., & Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. Knowledge-Based Systems, 189, 105124.
[11] Thakkar, A., & Lohiya, R. (2021). A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges. Archives of Computational Methods in Engineering, 28(4), 3211-3243.
[12] Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. Electronic Markets, 31(3), 685-695.
[13] Peng, J., Jury, E. C., Dönnes, P., & Ciurtin, C. (2021). Machine learning techniques for personalised medicine approaches in immune-mediated chronic inflammatory diseases: applications and challenges. Frontiers in Pharmacology, 2667.
[14] Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., & Ng, A. (2020). Cybersecurity data science: an overview from machine learning perspective. Journal of Big data, 7(1), 1-29.
[15] Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. A. (2020). Performance analysis of machine learning algorithms in intrusion detection system: a review. Procedia Computer Science, 171, 1251-1260.
[16] Ren, J., Guo, J., Qian, W., Yuan, H., Hao, X., & Jingjing, H. (2019). Building an effective intrusion detection system by using hybrid data optimization based on machine learning algorithms. Security and communication networks, 2019.
[17] Sultana, N., Chilamkurti, N., Peng, W., & Alhadad, R. (2019). Survey on SDN based network intrusion detection system using machine learning approaches. Peer-to-Peer Networking and Applications, 12(2), 493-501.
[18] Tahri, R., Balouki, Y., Jarrar, A., & Lasbahani, A. (2022). Intrusion Detection System Using machine learning Algorithms. In ITM Web of Conferences (Vol. 46, p. 02003). EDP Sciences.
[19] Carneiro, T., Da Nóbrega, R. V. M., Nepomuceno, T., Bian, G. B., De Albuquerque, V. H. C., & Reboucas Filho, P. P. (2018). Performance analysis of google colaboratory as a tool for accelerating deep learning applications. IEEE Access, 6, 61677-61685.
[20] Phan, T. N., Kuch, V., & Lehnert, L. W. (2020). Land Cover Classification using Google Earth Engine and Random Forest Classifier—The Role of Image Composition. Remote Sensing, 12(15), 2411.
[21] Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, 2(01), 20-28.
[22] Patel, H. H., & Prajapati, P. (2018). Study and analysis of decision tree based classification algorithms. International Journal of Computer Sciences and Engineering, 6(10), 74-78.
[23] Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In Automated machine learning (pp. 3-33). Springer, Cham.
[24] Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, 295-316.