Indones. J. Math. Nat. Sci. 46(2) 2023
Indonesian Journal of Mathematics and Natural Sciences
http://journal.unnes.ac.id/nju/index.php/JM

Optimalisasi Deteksi Serangan DDoS Menggunakan Algoritma Random Forest, SVM, KNN dan MLP pada Jaringan Komputer

Ilham Maulana*, Alamsyah

Program Studi Teknik Informatika, FMIPA, Universitas Negeri Semarang, Indonesia
Gedung D5 Lt.2, Kampus Sekaran Gunungpati, Semarang 50229
E-mail: 1ilhammaulana05303005@students.unnes.ac.id, alamsyah@mail.unnes.ac.id

Diterima 10 Agustus 2023 Disetujui 20 September 2023 Dipublikasikan 16 Oktober 2023

Abstrak

Distributed denial of service (DDoS) merupakan serangan pada server komputer yang menjadi ancaman serius pada keamanan jaringan komputer. Serangan ini dapat menyebabkan server komputer menjadi down. Untuk mengantisipasi serangan ini secara dini, dapat digunakan berbagai macam metode. Pada penelitian ini digunakan beberapa model algoritma *machine learning* yaitu *random forest, support vector machine, K-nearest neighbor* dan *multi-layer perceptron.* Hasil penelitian menunjukkan bahwa algoritma *random forest* memiliki akurasi sebesar 99,41%, *support vector machine* menghasilkan akurasi 98,37%, K–*nearest neighbor* menghasilkan akurasi 99%, dan *multi-layer perceptron* menghasilkan akurasi 93,97%. Algoritma *random forest* merupakan metode yang diusulkan terpilih dengan akurasi tertinggi yaitu 99,41%.

Kata kunci: *Random Forest*, SVM, SVM, MLP, DDoS attack

Abstract

Distributed denial of service is an attack on a computer server which poses a serious threat to computer network security. This attack can cause the computer server to go down. To anticipate this attack early, various methods can be used. In this research, several machine learning algorithm models were used, i.e., random forest, support vector machine, K–nearest neighbor and multi-layer perceptron. The research results show that the random forest algorithm has an accuracy of 99.41%, support vector machine produces an accuracy of 98.37%, K–nearest neighbor produces an accuracy of 99%, and multi-layer perceptron produces an accuracy of 93.97%. The random forest algorithm is the selected proposed method with the highest accuracy, i.e., 99.41%.

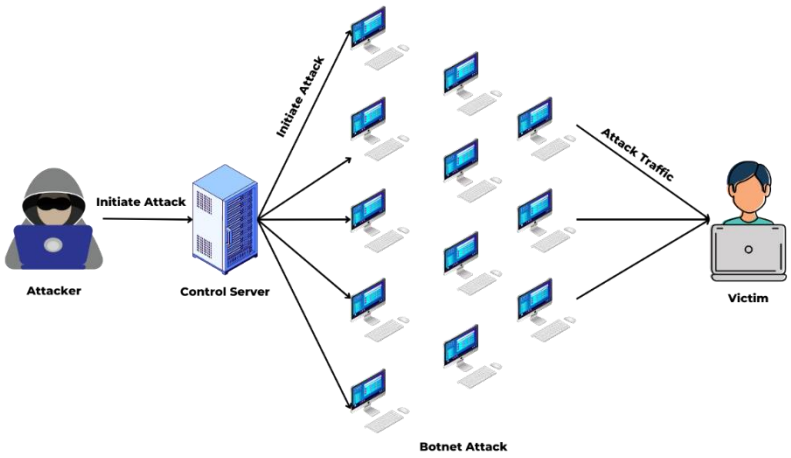
Keywords: *Random Forest*, SVM, SVM, MLP, DDoS attack

How to cite:
Maulana I., Alamsyah. (2023). Optimalisasi deteksi serangan DDoS menggunakan algoritma random forest, SVM, KNN dan MLP pada jaringan komputer. *Indonesian Journal of Mathematics and Natural Sciences*, 46(2), 83-92.

PENDAHULUAN

Serangan *distributed denial of service* (DDoS) merupakan sebuah serangan keamanan pada jaringan komputer yang sering kali dijumpai sejak tahun 1999 (Wang *et al*., 2020). Cara kerja serangan DDoS yakni dengan mengirimkan sebuah perintah atau sebuah permintaan dalam jumlah yang besar kepada satu server yang dikirimkan dari komputer yang berbeda. Serangan ini dikendalikan oleh satu komputer atau pusat perintah, sehingga mengakibatkan server menjadi *down* atau tidak bisa memberikan sebuah layanan. Server yang *down* tidak dapat melayani pengguna karena sudah terlalu banyak menerima permintaan (Tuan *et al*., 2020; Makuvaza *et al*., 2021). Dengan berkembangnya teknologi kecerdasan buatan (Prasetiyo *et al*, 2020; Walid & Alamsyah, 2017; Alamsyah & Fadila, 2021; Alamsyah *et al*., 2021) ancaman ini akan terus meningkat (Agarwal *et al*., 2022). Ketika terdapat aktivitas serangan yang dihasilkan oleh beberapa sumber, dan merupakan sebuah pesan spam maka

itu dikenal dengan serangan DDoS. Serangan DDoS bergerak secara bebas di jaringan internet, serangan ini melalui beberapa mesin perantara salah satunya yaitu dengan membentuk pasukan zombie yang biasa disebut sebagai botnet (Dora & Lakshmi, 2022) sebagaimana disajikan pada Gambar 1.



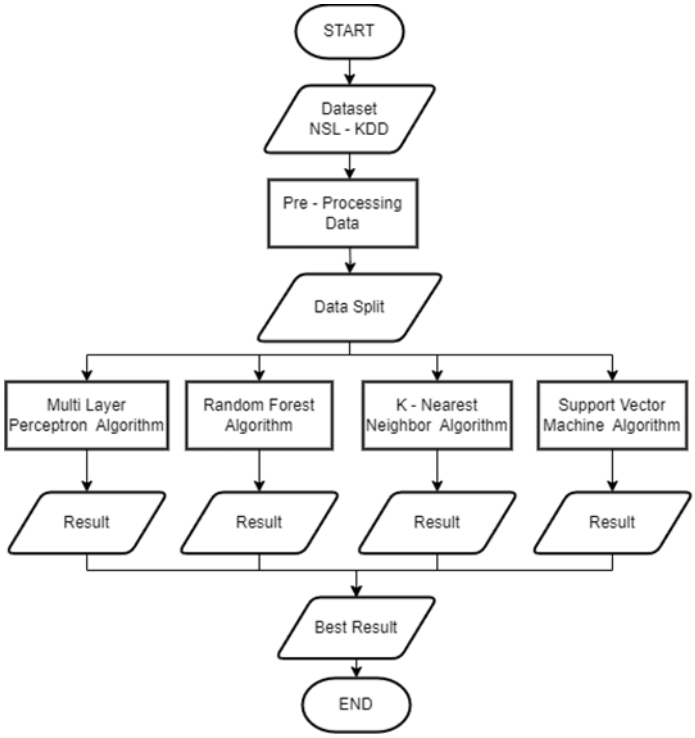
Gambar 1. Cara kerja serangan DDoS

Serangan DDoS dibagi menjadi dua kategori yaitu serangan yang menggunakan sumber daya sistem dan serangan yang menggunakan server botnet yang banyak untuk mendapatkan lalu lintas yang cepat agar menyatu dengan server target dan sepenuhnya menghabiskan sumber daya *bandwidth* jaringannya (Sivanesan & Archana, 2023; Aamir & Zaidi, 2019; Pajila *et al*., 2022).

Serangan DDoS dapat dideteksi secara dini dengan menggunakan machine learning diantaranya algoritma *random forest, K-nearest neighbor, support vector machine*, dan *multilayer perceptron*. Pada penelitian sebelumnya telah menghasilkan akurasi yang cukup signifikan yaitu pada algoritma *random forest* sebesar 97%, algoritma K-*nearest neighbor* menghasilkan akurasi 77,25%, algoritma *support vector machine* menghasilkan akurasi 78,12%, dan algoritma *multilayer perceptron* menghasilkan 74% (Najar & Naik S, 2022). Dengan hasil akurasi pada penelitian sebelumnya sudah cukup baik namun pada penelitian ini akan dikembangkan algoritma yang pernah diteliti sebelumnya sehingga dapat menghasilkan akurasi yang lebih tinggi dalam mendeteksi serangan DDoS pada jaringan komputer.

METODE

Alur dari metode yang diusulkan pada penelitian ini adalah dengan melakukan beberapa metode algoritma untuk mendeteksi serangan. Pada Gambar 2 disajikan langkah-langkah dari metode yang diusulkan. Pada penelitian ini menggunakan dataset NSL-KDD (Zaib, 2019). Pada dataset tersebut terdapat empat kelas serangan yang berbeda yakni: Serangan DoS/DDoS, Serangan Probe, Serangan Privilege, dan Serangan jaringan Akses (Najar & Naik S, 2022). Data ini berisikan informasi sebuah serangan yang berguna untuk mendeteksi anomali pada jaringan. Total dataset yang digunakan pada penelitian ini adalah 148.515. Data terbagi menjadi 2 bagian yaitu dataset normal sebanyak 77.053 dan dataset serangan DDoS sebanyak 71.462. Data latih untuk dataset normal berjumlah 67.342 dan serangan DDoS berjumlah 58.630. Sedangkan untuk data uji dataset normal berjumlah 9.711 dan serangan DDoS 12.832.



Gambar 2. Metode yang diusulkan

Pra-Processing Data

Pra-processing data dilakukan melalui langkah pelabelan biner, encoding fitur, dan persiapan data target. Pelabelan biner dilakukan dengan mengubah dataset dari string menjadi bentuk biner yaitu jika 'normal' maka akan diubah menjadi 0, jika 'serangan' maka akan diubah menjadi 1. Hasil akan disimpan dengan definisi *'is_attack'* dan lakukan juga untuk dataset test. Selanjutnya membuat sebuah tabel baru dengan mendefinisikan *'attack_flag'* yang berisikan hasil dari pemetaan serangan sebelumnya yang disimpan pada *'is_attack'*. Selanjutnya dilakukan klasifikasi jenis serangan ada pada dataset yaitu serangan *denial of service* (dos), serangan *probe,* serangan *privilege*, dan serangan access. Pembuatan plotting untuk kategori serangan dengan mendefinisikan nama *'attack_labels'* yang berisikan 'Normal','DoS',*'Probe','Privilege'*, dan *'Access*'. Dilakukan pemetaan pada jenis serangan dalam bentuk numerik, jika sebuah serangan tidak termasuk kedalam kategori yang ada pada *'attack_labels'* maka akan diberikan nilai 0. Nilai 0 adalah sebuah serangan normal. Hasil pemetaan disimpan dengan definisi *'map_attack'*, pengaplikasian pada fungsi *'map_attack'* hasil akan disimpan pada 'attack_map'. Penambahan kolom baru dengan nama *'attack_map'* yang berisikan hasil pemetaan serangan.

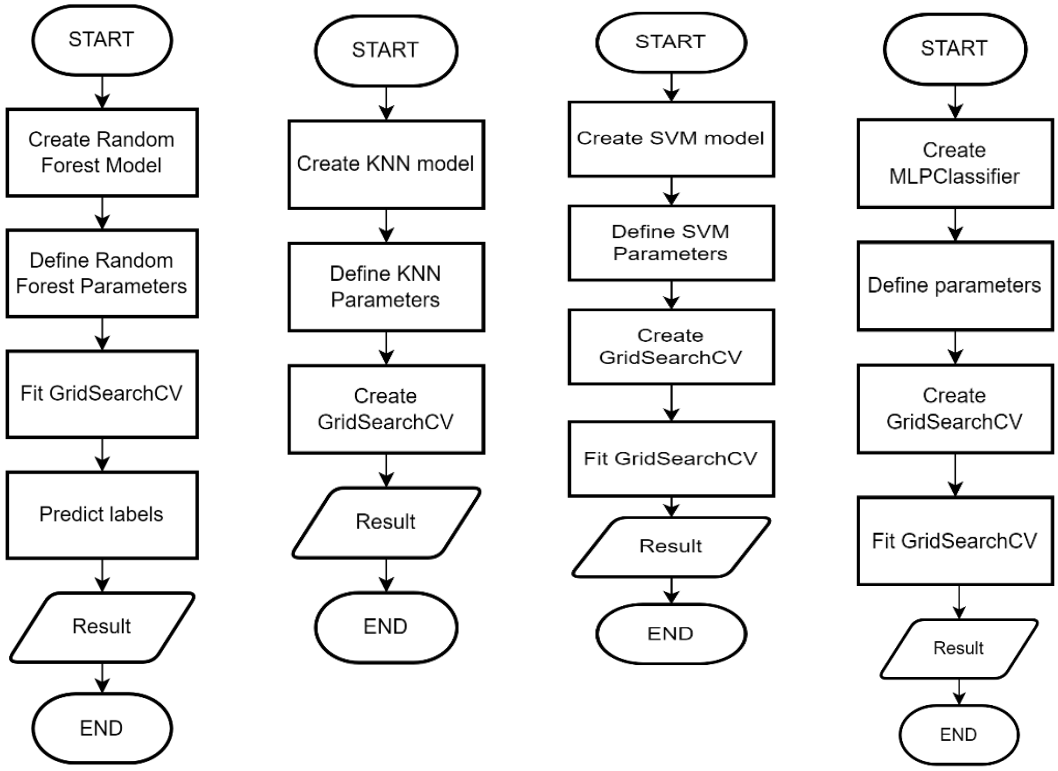
Encoding fitur diguunakan untuk mendefinisikan *'features_to_encode'* berisikan fitur *'protocol_type', 'service'*, dan *'flag*' akan dilakukan proses diencode yang berada pada data train dan test. Dengan menggunakan fungsi *'pd.get_dummies ()*' dapat dilakukan *one-hot encoding* pada fitur tersebut. Dilakukan proses penggabungan data train dan data test yang sudah dilakukan encode, lalu dilakukan proses pengurutan kolom kolom sesuai *'column_order'* disimpan dengan nama *'test_final'*. Selanjutnya mengambil fitur numerik yaitu *'duration', 'src_bytes'*, dan *'dst_bytes'*. Fitur ini diambil pada dataset train dan test, lalu digabungkan hasil encoding sebelumnya dan hasil didefinisikan dengan *'to_fit'* dan *'test_to'* yang siap untuk digunakan pemodelan.

Persiapan data target dilakukan dengan membuat sebuah 2 target yaitu untuk klasifikasi biner dan klasifikasi multi kelas. Target klasifikasi biner didefinisikan dengan *'binary_y'* yang berisikan *'attack_flag'*. Target klasifikasi multikelas didefinisikan dengan *'multi_y'* yang berisikan *'attack_map'*. Selanjutnya dilakukan inisialisasi untuk dataset train dan test. Dilakukan proses pemisahan dataset menjadi set pelatihan dan set validasi untuk klasifikasi biner dengan menggunakan fungsi *'train_test_split'* yang akan digunakan untuk membagi fitur (to_fit) dan targetnya *'binary_y'* dibagi menjadi empat subset yaitu *'binary_train_X* (fitur pelatihan)', *'binary_val_X*

(fitur validasi)', *'binary_train_y* (target pelatihan)', dan *'binary_val_y* (target validasi) '. Untuk klasifikasi multikelas sama halnya dengan klasifikasi biner yaitu dengan menggunakan *'train_test_split'* yang akan digunakan untuk membagi fitur (to_fit) dan targetnya *'binary_y'* dibagi menjadi empat subset yaitu *'multi_train_X* (fitur pelatihan)', *'multi_val_X* (fitur validasi)', *'multi_train_y* (target pelatihan)', dan *'multi_val_y* (target validasi)'.

Data Split

Pada tahap ini adalah sebuah tahapan yang dimana dataset yang sudah dilakukan proses *cleaning*, normalisasi dan pemilihan fitur akan dilakukan pemisahan data. Pada tahap ini dilakukan pemisahan data yakni 70% untuk data train dan 30% untuk data validation. Alur dari pembuatan *modelling* menggunakan algoritma *Random Forest*, *k – nearest neighbor, support vector machine,* da*n multi-layer perceptron* dapat dilihat pada Gambar 3.



Gambar 3. Flowchart dari algoritma *random forest* (A), *K – nearest neighbor* (B), *support vector machine* (C), da*n multi-layer perceptro* (D).

Random forest

Create random forest model digunakan untuk membangun model dengan menggunakan algoritma *random forest* dengan mengimport class sly yaitu *numpy, matlab*, dan *sklearn. ensemble*. Dengan menggunakan ensemble dapat menggabungkan sejumlah pohon keputusan (*decision tree*) untuk menghasilkan prediksi. Prediksi didapat melalui pengambilan hasil rata rata prediksi dari semua pohon dalam *random forest*, setiap pohon diberi bobot yang sama (Murugan *et al*., 2019).

Define random forest parameters digunakan untuk memberikan penjelasan terkait parameter yang akan digunakan pada model *Random Forest*. Model ini terdiri dari jumlah pohon (*n_estimator*), kedalam maksimum pada setiap pohon (*max_depth*), jumlah maksimum fitur yang digunakan dalam setiap split (*max_features*) dan parameter lain yang sering digunakan. Untuk mendapatkan model performa yang lebih optimal digunakan parameter yang sudah diidentifikasikan pada *pra – processing* (Khanday & Dadvandipour, 2021). Pada pembuatan parameters didefinisikan *'param_random'* yang berisikan *{'n_estimators': [3, 4], 'max_depth': [3, 6, None], 'bootstrap': [True, False]}*.

Create gridsearchCV digunakan untuk membuat sebuah operasi *GridSearchCV* sehingga dapat menemukan sebuah kombinasi parameter yang ideal untuk model. Pada setiap kombinasi parameter, *GridSearchCV* ini akan melakukan kinerja model dengan mengukur ukuran seperti akurasi. Pengaturan parameter yang lebih optimal dibutuhkan untuk mendapatkan kinerja model yang terbaik dengan menggunakan pencarian grid. Parameter yang akan dioperasikan *'param_random*, dengan jumlah *fold cross validation* '*cv*=3', dan melakukan evaluasi score menggunakan *'scoring = 'accuracy'*. Rumus yang digunakan disajikan pada Persamaan (1).

$$Accuracy = \frac{jumlah\ prediksi\ benar}{jumlah\ total\ prediksi} \quad (1)$$

Fit GridSearchCV digunakan untuk operasi *GridSearchCV* yang telah dibuat sebelumnya untuk dilatih modelling dengan menggunakan dataset train. Dalam menentukan parameter yang memberikan hasil performa terbaik, dengan menggunakan *GridSearchCV* akan melatih model dengan menggunakan setiap kombinasi parameter yang diberikan. *Predict labels* digunakan setelah pembuatan *GridSearchCV*, dilakukan prediksi label pada data uji untuk digunakan model. Dengan menggunakan model tersebut, maka dapat memprediksi data baru setelah melatih model dengan parameter yang lebih optimal. Hasil akan disimpan pada label atau inisialisasi lainnya yang mengantisipasi target yang dituju. *Confusion matriks* pada algoritma *random forest* digunakan untuk evaluasi hasil kinerja model yang diusulkan. Perhitungan *Confusion Matriks* disajikan pada Persamaan (2), (3), (4), dan (5).

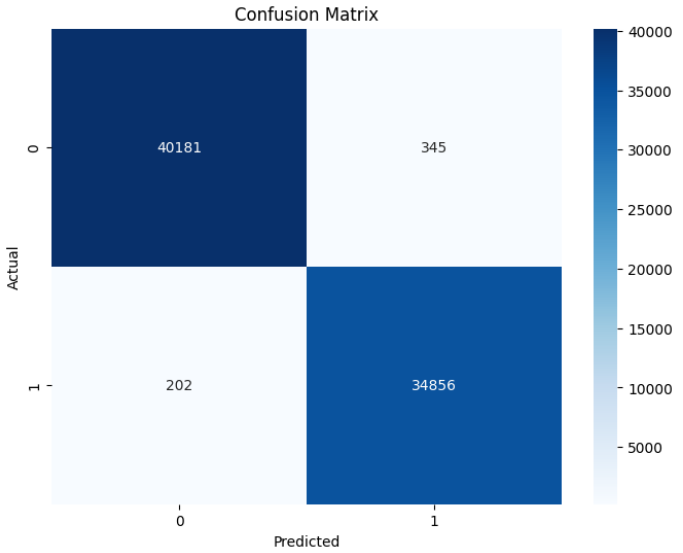
$$Accuracy\ Value = \frac{TP+TN}{TP+FP+FN+TN} \times 100 \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

$$F1\ Score = 2 \times \frac{Precision \times recall}{Precision+recall} \quad (5)$$

Di mana menurut Ahmadi *et al*. (2018), *True Negative* (TN) adalah jumlah data yang diklasifikasikan dengan benar sebagai negatif (kelas negatif) oleh model. Pada kasus ini, terdapat 40.181 data yang diklasifikasikan dengan benar sebagai negatif. *False Positive* (FP) adalah jumlah data yang salah diklasifikasikan sebagai positif (kelas positif) oleh model. Pada kasus ini, terdapat 345 data yang salah diklasifikasikan sebagai positif. *False Negative* (FN) adalah jumlah data yang salah diklasifikasikan sebagai negatif oleh model. Pada kasus ini, terdapat 202 data yang salah diklasifikasikan sebagai negatif. *True Positive* (TP) adalah jumlah data yang diklasifikasikan dengan benar sebagai positif oleh model. Pada kasus ini, terdapat 34.856 data yang diklasifikasikan dengan benar sebagai positif. Gambar 4 menyajikan model perhitungan *Confusion Matriks*.



Gambar 4. *Confusion Matriks*

K – Nearest Neighbor

KNN adalah teknik *machine learning* yang dasar dan mudah untuk diimplementasikan (Mihoub *et al.*, 2022). Algoritma ini digunakan untuk metode klasifikasi yang menggunakan jarak antara titik data untuk memprediksi label dari data yang belum diketahui (Mihoub *et al*., 2022; Wazirali, 2020). Pada KNN, label data baru ditentukan berdasarkan mayoritas label dari *K* terdekatnya (Wazirali, 2020). Parameter-parameter yang akan digunakan dalam model KNN didefinisikan. Beberapa parameter yang umum digunakan adalah jumlah tetangga terdekat yang akan digunakan untuk menentukan label *(n_neighbors),* metode penentuan bobot tetangga *(weights*), dan parameter jarak. Parameter yang akan dioptimasi akan didefinisikan *'param_knn'* yang berisikan '*n_neighbors'* (jumlah tetangga terdekat) berjumlah [3, 5, 7], *'weights'* (metode pembobotan) menggunakan [*'uniform', 'distance'*], dan 'p' (parameter jarak) berjarak [1, 2].

Pada tahap *create GridSearchCV*, objek GridSearchCV untuk model KNN dibuat. GridSearchCV digunakan untuk mencari kombinasi parameter terbaik untuk model KNN. Dengan melakukan pencarian grid, dapat ditemukan kombinasi parameter terbaik yang menghasilkan performa model yang optimal. Rumus perhitungan scoring dapat dilihat pada rumus (5). Parameter yang akan dioptimasikan *'param_knn'*, dengan jumlah *fold cross validation 'cv=3'*, dan melakukan evaluasi score menggunakan *'scoring = 'accuracy'*. Proses perhitungan akurasi terjadi melalui *3-fold cross validation* yang dilakukan oleh objek *GridSearchCV*.

Pada tahap *Fit GridSearchCV,* model menggunakan data latih dengan menggunakan objek *GridSearchCV* yang telah dibuat sebelumnya. *GridSearchCV* akan melakukan pelatihan model pada setiap kombinasi parameter yang ditentukan dan mencari parameter terbaik yang menghasilkan performa model terbaik (Nobakht *et al*., 2022). Pelatihan model KNN melibatkan membangun struktur data yang memungkinkan pencarian tetangga terdekat dengan cepat.

Support Vector Machine

SVM adalah sebuah model yang digunakan untuk analisis regresi dan klasifikasi. SVM digunakan untuk masalah klasifikasi biner dan untuk masalah klasifikasi multikelas. SVM menggunakan titik data pelatihan terdekat dari setiap data untuk membangun satu atau lebih pada *hyperplane* untuk mengklasifikasikan data berdimensi tinggi. SVM terdapat satu kelas mencoba menghitung *hyperplane* yang sebagian kecil dari sampel data latih terdekat dari setiap data untuk membangun *hyperplane* untuk mengklasifikasikan data berdimensi tinggi (Hosseinzadeh *et al*., 2021).

Pada tahap *define SVM parameters,* didefinisikan parameter-parameter yang digunakan dalam model SVM. Parameter akan dilakukan optimasi pada mode SVM didefinisikan dalam bentuk dictionary *'param_svm'* berisikan {'*C*': [0.1, 1, 10], 'gamma': [0.1, 1, '*scale*']}, parameter yang dilakukan optimasi meliputi 'c' sebagai penalty parameter dan *'gamma'* sebagai parameter kernel. Membuat objek *GridSearchCV* untuk model SVM. *GridSearchCV* digunakan untuk mencari kombinasi parameter terbaik untuk model SVM. Dengan melakukan pencarian grid, dan mencari kombinasi parameter terbaik yang menghasilkan performa model yang optimal. Data latih akan dimasukan dalam objek yang dibuat untuk *GridSearchCV*. Dalam pembuatan *GridSearchCV* akan melakukan optimasi pada parameter *'param_svm'*, dengan jumlah *fold cross validation* '*cv*=3', dan melakukan evaluasi score menggunakan *'scoring = 'accuracy'*. Proses perhitungan akurasi terjadi melalui *3-fold cross validation* yang dilakukan oleh obyek *GridSearchCV*. Melatih model menggunakan data latih dengan menggunakan objek *GridSearchCV* yang telah dibuat sebelumnya. *GridSearchCV* akan melakukan pelatihan model pada setiap kombinasi parameter yang ditentukan dan mencari parameter terbaik yang menghasilkan performa model terbaik (Srihari et al., 2023). Pelatihan model SVM melibatkan mencari hyperplane optimal yang memaksimalkan margin dan meminimalkan kesalahan klasifikasi.

Multi-Layer Perceptron

MLP adalah sebuah jaringan saraf tiruan yang memiliki beberapa lapisan input, lapisan output, dan lapisan tertutup. Setiap *neuron* menerima input dari *neuron-neuron* pada lapisan sebelumnya dan menghasilkan output yang kemudian diteruskan ke lapisan berikutnya. MLP digunakan untuk masalah klasifikasi dan regresi (Farzad & Gulliver, 2022). Pembuatan model MLP dengan menggunakan *solver 'lbfgs; solver* ini digunakan untuk optimasi dan *random_state* diatur menjadi 1.

Parameter-parameter yang akan digunakan didefinisikan dalam model MLP. Beberapa parameter yang umum digunakan adalah jumlah lapisan dan jumlah neuron dalam setiap lapisan, fungsi aktivasi yang digunakan oleh setiap neuron, dan metode optimasi yang digunakan untuk

melatih model. Penelitian ini mendefinisikan dalam bentuk dictionary *'params_mlp'*, dengan parameter yang dioptimasi yaitu *'alpha'* (parameter untuk regularisasi) dan *'hidden_layer_sizes'* (jumlah dan ukuran lapisan tersembunyi). Pada *params_mlp* berisikan pada *'alpha':* [1*e*-5, 1*e*-3, 1*e*-1], dan untuk *'hidden_layer_sizes':* [(5, 2), (10, 5), (20, 10)]. Tahap berikutnya adalah membuat objek *GridSearchCV* untuk model MLP. *GridSearchCV* digunakan untuk mencari kombinasi parameter terbaik untuk model MLP. Dengan melakukan pencarian grid, dan mencari kombinasi parameter terbaik yang menghasilkan performa model yang optimal (Srihari *et al*., 2023). Data latih akan dimasukan dalam objek yang dibuat untuk *GridSearchCV*. Parameter yang di optimasi yaitu pada *'params_mlp'*, dengan menggunakan jumlah *fold cross-validation 'cv = 7'* dan untuk scoring yang digunakan untuk evaluasi *'scoring = accuracy'*. Proses perhitungan akurasi terjadi melalui *7-fold cross validation* yang dilakukan oleh objek *GridSearchCV*. Pelatihan model menggunakan data latih dengan menggunakan objek *GridSearchCV* yang telah dibuat sebelumnya. *GridSearchCV* akan melakukan pelatihan model pada setiap kombinasi parameter yang ditentukan dan mencari parameter terbaik yang menghasilkan performa model terbaik. Pelatihan model MLP melibatkan proses *backpropagation* yang mengoptimalkan bobot dan bias jaringan untuk meminimalkan fungsi loss (Rumelhart & McClelland, 1987). Hasil keluaran dari lapisan tersembunyi disajikan pada Persamaan (6) dan (7).

$$h = activation\_function\ (Weight\_matrix_{input} + vector\ bias) \quad (6)$$

$$y = activation\_function\ (Weight\_matrix_{h} + vector\ bias) \quad (7)$$

HASIL DAN PEMBAHASAN

Setelah dilakukannya beberapa pengimplementasian pengklasifikasi *machine learning* untuk berbagai macam metode, klasifikasi menggunakan *Random Forest* memiliki kinerja yang lebih optimal dibandingkan dengan metode lainnya seperti yang disajikan pada Tabel 1. *Random Forest* dilatih dengan 70% untuk data pelatihan dan untuk 30% pada data validasi. Penelitian ini menggunakan 8 *columns* dari dataset yang terdapat 43 *columns*, Dengan menggunakan metode *GridSearchCV* menghasilkan akurasi yang lebih optimal. Parameter setiap model akan dilakukan optimasi dengan jumlah *fold cross validation* yang berbeda beda dan dilakukan evaluasi score menggunakan *'scoring='accuracy'*. Dengan menggunakan algoritma *Random Forest* untuk parameter akan dilakukan optimasi dengan oleh *GridSearchCV*, dengan jumlah *fold cross validation* sebesar 3, artinya perhitungan akurasi terjadi sebanyak *3-fold cross validation* yang dilakukan oleh *GridSearchCV*, memberikan nilai hasil akurasi 99,42% pada akurasi *train*, 99.36% pada akurasi *test*, dan 99.41% pada akurasi keseluruhan. Dengan menggunakan algoritma *K -Nearest Neighbor* untuk parameter akan dilakukan optimasi dengan oleh *GridSearchCV*, dengan *jumlah fold cross validation* sebesar 3, artinya perhitungan akurasi terjadi sebanyak *3-fold cross validation* yang dilakukan oleh *GridSearchCV.* memberikan nilai hasil akurasi 99% pada akurasi *train*, 99% pada akurasi *test*, dan 99% pada akurasi keseluruhan. Dengan menggunakan algoritma *Support Vector Machine* untuk parameter akan dilakukan optimasi dengan oleh *GridSearchCV*, dengan jumlah *fold cross validation* sebesar 3, artinya perhitungan akurasi terjadi sebanyak *3-fold cross validation* yang dilakukan oleh *GridSearchCV*, memberikan nilai hasil akurasi 99,43% pada akurasi *train*, 98,37% pada akurasi test, dan 98,37% pada akurasi keseluruhan. Dengan menggunakan algoritma *Multi-Layer Perceptron* untuk parameter akan dilakukan optimasi dengan oleh *GridSearchCV*, dengan jumlah *fold cross validation* sebesar 7, artinya perhitungan akurasi terjadi sebanyak 3-fold *cross* validasi yang dilakukan oleh *GridSearchCV*, memberikan nilai hasil akurasi 93,93% pada akurasi *train*, 93,97% pada akurasi *test*, dan 93.97% pada akurasi keseluruhan.

Tabel 1. *Accuracy comparison of different models*

	Models			
	RF	MLP	SVM	KNN
Train Accuracy	0,9942	0,9393	0,9943	0,99
Test Accuracy	0,9936	0,9397	0,9837	0,99
Accuracy	0,9941	0,9397	0,9837	0,99
Precision	0,9910	0,9848	0,98	0,99
Recall	0,9953	0,8857	0,98	0,99
F1 Score	0,9931	0,9326	0,98	0,99

Pada Tabel 2 menyajikan perbandingan hasil dari metode yang diusulkan dengan hasil penelitian-penelitian sebelumnya. Berdasarkan Tabel 2, hasil penelitian dari metode yang diusulkan memiliki akurasi 99,41%. Hasil penelitian yang diusulkan memiliki tingkat akurasi yang lebih tinggi dari peneliti-peneliti sebelumnya. Hasil akurasi dari peneliti-peneliti sebelumnya yaitu Imamverdiyev dan Abdullayeva (2018) dengan akurasi 73,23%, Tang *et al*. (2019) dengan akurasi 88,39%, Rusyaidi *et al*. (2022) dengan akurasi 88,39, Aslan (2022) dengan akurasi 96,6%, Liu *et al*. (2021) dengan akurasi 85,24, Ugwu *et al*. (2021) dengan akurasi 86%, dan Najar dan Naik (2022) dengan akurasi 97% (Tabel 2).

Tabel 2. Perbandingan akurasi

Peneliti	Akurasi
Imamverdiyev & Abdullayeva (2018)	73,23%
Tang *et al.* (2019)	88,39%
Rusyaidi *et al*. (2022)	88,39%
Aslan (2022)	96,60%
Liu *et al*. (2021)	85,24%
Ugwu *et al*. (2021)	86,00%
Najar & Naik (2022)	97,00%
Metode yang diusulkan	99,41%

SIMPULAN

Beberapa model *machine learning* digunakan dalam penelitian ini dalam mendeteksi sebuah inputan atau paket berupa serangan atau normal. Dengan mencoba beberapa model machine learning terdapat sebuah akurasi yang lebih optimal yaitu *random forest*. Metode ini menggunakan GridSearchCV untuk mencari parameter terbaik atau kombinasi parameter agar mendapatkan hasil yang lebih optimal. Hasil penelitian menunjukkan 99,42% pada data latih, 99,37% pada data tes, dan 99,41% pada data uji dengan nilai f1 sebesar 99,32%. Penelitian ini juga menggunakan model lain yaitu KNN, SVM, MLP. Hasil terbaik akurasi sebesar 99,41%. Hasil ini lebih baik dibandingkan dengan hasil penelitian-penelitian sebelumnya. Setelah memberikan hasil yang lebih optimal, diharapkan metode yang diusulkan dapat bermanfaat untuk digunakan dalam mekanisme keamanan pada jaringan komputer untuk mendeteksi aktivitas mencurigakan.

DAFTAR PUSTAKA

Aamir, M., & Zaidi, S. M. A. (2019). DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation. *International Journal of Information Security*, 18(6), 761-785. https://doi.org/10.1007/s10207-019-00434-1

Agarwal, A., Khari, M., & Singh, R. (2022). Detection of DDOS attack using deep learning model in cloud storage application. *Wireless Personal Communications*, 127(1), 419-439. https://doi.org/10.1007/s11277-021-08271-z

Ahmadi, E., Weckman, G. R., & Masel, D. T. (2018). Decision making model to predict presence of coronary artery disease using neural network and C5.0 decision tree. *Journal of Ambient Intelligence and Humanized Computing*, 9(4), 999-1011. https://doi.org/10.1007/s12652-017-0499-z

Alamsyah, A., & Fadila, T. (2021). Increased accuracy of prediction hepatitis disease using the application of principal component analysis on a support vector machine. *Journal of Physics: Conference Serie*s, 1968 012016. https://doi.org/10.1088/1742-6596/1968/1/012016

Alamsyah, A., Prasetiyo, B., Hakim, M., & Pradana, F. (2021). Prediction of COVID-19 using recurrent neural network model. *Scientific Journal of Informatics*, 8(1), 98-103. doi:https://doi.org/10.15294/sji.v8i1.30070

Aslan, O. (2022). Using machine learning techniques to detect attacks in computer networks. *Conference: Aegean Summit 4th International Applied Sciences CongressAt: Mugla, Turkey*.

Dora, V. R. S., & Lakshmi, V. N. (2022). Optimal feature selection with CNN-feature learning for DDoS attack detection using meta-heuristic-based LSTM. *International Journal of Intelligent Robotics and Applications*, 6(2), 323-349. https://doi.org/10.1007/s41315-022-00224-4

Farzad, A., & Gulliver, T. A. (2022). Log message anomaly detection with fuzzy C-means and MLP. *Applied Intelligence*, 52(15), 17708-17717. https://doi.org/10.1007/s10489-022-03300-1

Hosseinzadeh, M., Rahmani, A. M., Vo, B., Bidaki, M., Masdari, M., & Zangakani, M. (2021). Improving security using SVM-based anomaly detection: issues and challenges. *Soft Computing*, 25(4), 3195-3223. https://doi.org/10.1007/s00500-020-05373-x

Imamverdiyev, Y., & Abdullayeva, F. (2018). Deep learning method for denial of service attack detection based on restricted boltzmann machine. *Big Data*, 6(2), 159-169. https://doi.org/10.1089/big.2018.0023

Khanday, O. M., & Dadvandipour, S. (2021). Analysis of machine learning algorithms for character recognition: a case study on handwritten digit recognition. *Indonesian Journal of Electrical Engineering and Computer Science*, 21(1), 574-581. https://doi.org/10.11591/ijeecs.v21.i1.pp574-581

Liu, C., Gu, Z., & Wang, J. (2021). A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning. *IEEE Access*, 9, 75729-75740. https://doi.org/10.1109/ACCESS.2021.3082147

Makuvaza, A., Jat, D. S., & Gamundani, A. M. (2021). Deep neural network (DNN) solution for real-time detection of distributed denial of service (DDoS) attacks in software defined networks (SDNs). *SN Computer Science*, 2(2), 1-10. https://doi.org/10.1007/s42979-021-00467-1

Mihoub, A., Ben, F. O., Cheikhrouhou, O., Derhab, A., & Krichen, M. (2022). Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques. *Computers & Electrical Engineering*, 98, 107716. https://doi.org/10.1016/j.compeleceng.2022.107716

Murugan, A., Nair, S. A. H., & Kumar, K. P. S. (2019). Detection of skin cancer using SVM, random forest and kNN classifiers. *Journal of Medical Systems*, 43(8). https://doi.org/10.1007/s10916-019-1400-8

Najar, A., & Naik S. M. (2022). DDoS attack detection using MLP and random forest algorithms. *International Journal of Information Technology*, 14. https://doi.org/10.1007/s41870-022-01003-x

Nobakht, M., Javidan, R., & Pourebrahimi, A. (2022). DEMD-IoT: a deep ensemble model for IoT malware detection using CNNs and network traffic. *Evolving Systems*, 14, 1-17. https://doi.org/10.1007/s12530-022-09471-z

Pajila, P. J. B., Julie, E. G., & Robinson, Y. H. (2022). FBDR-fuzzy based DDoS attack detection and recovery mechanism for wireless sensor networks. *Wireless Personal Communications*, 122(4). https://doi.org/10.1007/s11277-021-09040-8

Prasetiyo, B., Alamsyah, A., Muslim, M.A., Subhan, S., & Baroroh, N. (2020). Artificial neural network model for banckrupty prediction. *Journal of Physics: Conference Series*, 1567 032022. https://doi.org/10.1088/1742-6596/1567/3/032022

Rumelhart, D. E., & McClelland, J. L. (1987). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, 318-362. MIT Press. http://ieeexplore.ieee.org/document/6302929

Rusyaidi, M., Jaf, S., & Ibrahim, Z. (2022). Detecting distributed denial of service in network traffic with deep learning. *International Journal of Advanced Computer Science and Applications*, 13(1), 34-41. https://doi.org/10.14569/IJACSA.2022.0130105

Sivanesan, N., & Archana, K. S. (2023). Detecting distributed denial of service (DDoS) in SD-IoT environment with enhanced firefly algorithm and convolution neural network. *Optical and Quantum Electronics*, 55(5), 1-16. https://doi.org/10.1007/s11082-023-04553-x

Srihari, P., Santosh, V., & Ganapathy, S. (2023). An epileptic seizures diagnosis system using feature selection, fuzzy temporal naive Bayes and T-CNN. *Multimedia Tools and Applications*, 34075–34094. https://doi.org/10.1007/s11042-023-14928-7

Tang, T. A., McLernon, D., Mhamdi, L., Zaidi, S. A. R., & Ghogho, M. (2019). Intrusion detection in SDN-based networks: deep recurrent neural network approach BT-deep learning applications for cyber security. M. Alazab & M. Tang (eds.), 175-195. Springer International Publishing. https://doi.org/10.1007/978-3-030-13057-2_8

Tuan, T. A., Long, H. V., Son, L. H., Kumar, R., Priyadarshini, I., & Son, N. T. K. (2020). Performance evaluation of botnet DDoS attack detection using machine learning. *Evolutionary Intelligence*, 13(2), 283-294. https://doi.org/10.1007/s12065-019-00310-w

Ugwu, C. C., Obe, O. O., Popoọla, O. S., & Adetunmbi, A. O. (2021). A distributed denial of service attack detection system using long short term memory with singular value decomposition. *IEEE*

2nd International Conference on Cyberspac (Cyber Nigeria), 112-118. https://doi.org/10.1109/CYBERNIGERIA51635.2021.9428870

Walid, & Alamsyah. 2017. Recurrent neural network for forecasting time series with long memory pattern. *Journal of Physics: Conference Series*, 824, 012038). https://doi.org/10.1088/1742-6596/824/1/012038

Wang, M., Lu, Y., & Qin, J. (2020). A dynamic MLP-based DDoS attack detection method using feature selection and feedback. *Computers and Security*, 88. https://doi.org/10.1016/j.cose.2019.101645

Wazirali, R. (2020). An improved intrusion detection system based on KNN hyperparameter tuning and cross-validation. *Arabian Journal for Science and Engineering*, 45(12), 10859-10873. https://doi.org/10.1007/s13369-020-04907-7

Zaib, H. M. (2019). Dataset NSL-KDD. https://www.kaggle.com/datasets/hassan06/nslkdd