Jurnal Masyarakat Informatika *ISSN: 2086-4930*
Vol. 14, No. 1, 2023 *e-ISSN: 2777-0648*

Intrusion Detection Systems pada Bot-IoT *Dataset* Menggunakan Algoritma *Machine Learning*

Jonathan Nicholas Sibarani[1)], Dheo Ronaldo Sirait[2)], dan Salma Safira Ramadhanti[3)]

Departemen Ilmu Komputer/Informatika, Fakultas Sains dan Matematika, Universitas Diponegoro
[1)]jonathan.nicholas.14052001@gmail.com, [2)]dheoronaldosirait@gmail.com,
[3)]salmasafira2111@gmail.com

Abstrak

Semakin berkembangnya dunia teknologi, semakin banyak juga penggunaan internet dalam kehidupan sehari hari. Pertumbuhan dalam penggunaan internet tersebut menimbulkan kekhawatiran tentang keamanan saat menggunakan layanan internet. Untuk menjamin keamanan pengguna, dapat menggunakan Intrusion Detection System (IDS). Intrusion Detection System merupakan sebuah sistem yang akan mengawasi aktivitas dalam jaringan komputer dengan menggunakan berbagai macam metode seperti machine learning. Dalam jurnal penelitian ini, digunakan tiga macam algoritma machine learning untuk membantu IDS dalam mengenali serangan. Algoritma machine learning yang digunakan adalah K-Nearest Neighbor, Random Forest, dan Gaussian Naïve Bayes. Untuk membantu penelitian juga digunakan BoT-IoT Dataset yang dibuat oleh UNSW Canberra dengan lebih dari 72.000.000 baris data. Penelitian ini dilakukan dengan tujuan untuk menentukan algoritma yang paling sesuai dalam melakukan deteksi intrusi dengan dataset BoT-IoT.

Kata kunci : *K-Nearest Neighbor, Random Forest, Gaussian Naive Bayes, Intrusion Detection System, Machine Learning, Cybersecurity*

Abstract

As technological advancement grows, so is the daily usage of internet. The growing usage of internet raises concerns about security safety when using services on the internet. To ensure user security, the Intrusion Detection System (IDS) can be used. Intrusion Detection System is a system that will monitor activities in a computer network using various methods such as machine learning. In this research journal, three kinds of machine learning algorithms are used to assist IDS in recognizing attacks. The machine learning algorithms used are K-Nearest Neighbor, Random Forest, and Gaussian Naïve Bayes. To assist the research, the BoT-IoT Dataset created by UNSW Canberra was also used by taking 5% of the entire dataset. This research was conducted with the aim of determining the most suitable algorithm in performing intrusion detection with the BoT-IoT dataset.

Keywords : *K-Nearest Neighbor, Random Forest, Gaussian Naive Bayes, Intrusion Detection System, Machine Learning, Cybersecurity*

1 PENDAHULUAN

Pada era global saat ini, bidang *Internet of Things* (IoT) telah berkembang sangat pesat. Terdapat lebih dari 20 miliar perangkat yang saling terhubung, termasuk komputer, berbagai perangkat pintar serta perangkat akses internet 4G dan 5G yang sangat cepat digunakan dalam berbagai jenis aplikasi, seperti edukasi, transportasi, kesehatan, dan sebagainya [1]. Sejumlah besar perangkat yang saling terhubung tersebut terus-menerus bertukar dan mengirimkan sejumlah besar data (*Big Data*) dan menjadikan sistem IoT sebagai target untuk berbagai jenis serangan siber. Sehingga sistem keamanan jaringan menjadi salah satu aspek yang penting.

Pada tahun 1987, Denning mengusulkan tentang mendeteksi dan mengidentifikasi adanya intrusi melalui IDS atau *Intrusion Detection System.* IDS merupakan sebuah sistem perangkat keras atau perangkat lunak yang dapat digunakan untuk mendeteksi adanya aktivitas yang mencurigakan dalam jaringan atau sistem komputer. IDS dikembangkan menjadi dua jenis yaitu *misuse detection* dan *anomaly detection*. *Misuse detection* memanfaatkan pencocokan pola berdasarkan *database* yang telah didefinisikan. Sedangkan, *anomaly detection* memanfaatkan ketidaknormalan dari aktivitas yang dilakukan pada jaringan jika dibandingkan dengan kondisi jaringan dalam keadaan normal [2]. *Intrusion Detection System* dapat membantu untuk mendeteksi serangan siber pada sistem IoT. Permasalahan IDS untuk mendeteksi serangan dapat didekati dengan berbagai algoritma *machine learning*.

Penelitian ini bertujuan untuk menerapkan, menganalisis, serta membandingkan tiga algoritma *machine learning* yang banyak digunakan saat ini yaitu *K-Nearest Neighbors*, *Random Forest*, dan *Gaussian Naive Bayes* dengan menggunakan BOT-IoT *dataset* yang memiliki tiga jenis label atau kelas yaitu *attack*, *category*, dan *subcategory*. Fitur dari *dataset* yang digunakan merupakan 10 fitur terbaik yang ditetapkan penulis *dataset*. Performa dari setiap algoritma yang diterapkan akan dinilai berdasarkan nilai *recall*, *precision*, *F1-score*, dan *accuracy*.

Penyusunan penulisan dilakukan sebagai berikut. Pada Bab 2 menjelaskan tinjauan pustaka yang berisi penelitian terkait *Intrusion Detection System*. Bab berikutnya menjelaskan mengenai metode penelitian, algoritma *K-Nearest Neighbors*, *Random Forest*, *Gaussian Naive Bayes*, serta *dataset* yang digunakan. Bab 4 menjelaskan hasil dan pembahasan dari percobaan yang telah dilakukan. Bab terakhir berisi kesimpulan dari hasil penelitian.

2 TINJAUAN PUSTAKA

Penelitian terkait Intrusion Detection Systems pada Bot-IoT Dataset Menggunakan Algoritma *Machine Learning* telah dilakukan menggunakan berbagai metode. Dalam melakukan penelitian ini, temuan harus diperkuat dengan melakukan tinjauan pustaka referensi, mengidentifikasi metode yang digunakan, dan mengembangkan penelitian sebelumnya yang menunjukkan korelasi yang seimbang. Beberapa tinjauan literatur yang telah mengalami metode serupa antara lain:

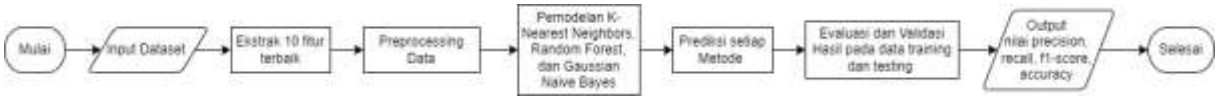
1. Penelitian yang dilakukan oleh (Koroniotis et al. 2019) [6] dengan judul "*Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset*". Tujuan dari penelitian ini adalah melakukan perlindungan realistis dan penanggulangan investigasi, seperti deteksi intrusi jaringan dan sistem

forensik jaringan, perlu dikembangkan secara efektif. Penelitian ini menggunakan pengklasifikasi akurat, dengan *Recurrent Neural Networks* (RNN) dan mereka permutasi (LSTM) mengungguli implementasi SVM dengan akurasi masing masing 0.997406 untuk RNN, 0.9974194 untuk LSTM dan SVM sebesar 0.8837 dengan menggunakan 10 fitur terbaik. Model dilatih pada versi 10 fitur terbaik yang dipilih dari *dataset* ditampilkan akurasi yang lebih tinggi daripada versi lengkap.

2. Penelitian yang dilakukan oleh (Pokhrel, Abbas, and Aryal 2021) [7] dengan judul "*IoT Security: Botnet detection in IoT using Machine learning*" dengan tujuan untuk melakukan Perbandingan performansi ketiga algoritma yang digunakan dilakukan pada dataset ketidakseimbangan kelas dan pada dataset keseimbangan kelas. Algoritma terbaik dipilih oleh titik referensi berdasarkan persentase akurasi dan area di bawah skor kurva karakteristik operasi penerima (ROC AUC). Penelitian ini kurasi 92,1% dan 92,2% ROC AUC dari algoritma KNN. Juga, algoritma KNN bekerja dengan baik pada data *real- time* yang sangat tidak seimbang. KNN efektif digunakan dalam sistem deteksi botnet. Di antara algoritma mesin yang berbeda, kami mendapatkan akurasi yang lebih tinggi dari algoritma KNN. Dari perbandingan keseluruhan pada metrik evaluasi yang berbeda dari algoritma pembelajaran mesin, algoritma KNN ditemukan sebagai yang terbaik untuk kumpulan data BoT-IoT.
3. Selanjutnya, penelitian yang dilakukan oleh (Shafiq et al. 2020) [8] dengan judul "*Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city*". Penelitian ini dilakukan dengan tujuan untuk mengetahui algoritma ML mana yang efektif dan harus digunakan untuk memilih anomali IoT dan identifikasi lalu lintas intrusi, diterapkan pendekatan *soft set bijective* dan algoritmanya. Kemudian kami menerapkan algoritma yang diusulkan berdasarkan pendekatan *soft set bijective*. Algoritma yang digunakan adalah *BayesNet* dengan akurasi 99.77*, C4.5* dengan akurasi 99.99*, NaiveBayes* dengan akurasi 99.79*, Random Forest* dengan akurasi 99.99*, dan Random Tree* dengan akurasi 99.99.
4. Penelitian ini menerapkan algoritma *K-Nearest Neighbors (KNN), Random Forest,* dan *Gaussian Naive Bayes.* Penerapan ketiga metode diharapkan bisa memberi hasil prediksi yang lebih baik dan mampu menangani *task* dalam jumlah data yang sangat besar.

3 METODE PENELITIAN

Untuk memudahkan penelitian agar dapat berjalan dengan sistematis serta memenuhi tujuan yang diinginkan dalam melakukan analisis berbagai jenis serangan menggunakan *dataset* BoT-IoT dan membandingkan hasilnya dengan berbagai algoritma klasifikasi *machine learning* pada *intrusion detection system* maka dibuat *flowchart* pada tahapan penelitian yang ditunjukan pada Gambar 1.



Gambar 1 ***Flowchart*** **Tahapan** ***Intrusio n Detection System***

Tahapan pertama yang dilakukan pada penelitian adalah memasukkan *dataset* yang digunakan, kemudian mengekstraksi 10 fitur terbaik dan label pada *dataset*. Setelahnya akan dilakukan *preprocessing* data, mendefinisikan kelas model, prediksi untuk setiap metode, dan mengevaluasi hasil pada data *training* dan *testing*. Keluaran hasil yang akan dianalisis yaitu nilai *precision*, *recall*, *f1-scores*, dan *accuracy*.

3.1 *INTRUSION DETECTION DATASETS (*BOT-IOT *DATASET*)

Penelitian ini menggunakan *dataset* BoT-IoT yang dibuat oleh UNSW Canberra dengan merancang lingkungan jaringan yang realistis di *Cyber Range Lab* UNSW Canberra. Lingkungan jaringan menggabungkan kombinasi lalu lintas normal dan botnet. *Dataset* BoT-IoT memiliki lebih dari 72.000.000 baris data. Untuk mempermudah penanganan *dataset*, UNSW Canberra mengekstrak 5% *dataset* asli melalui penggunaan *query* MySQL [3]. Penelitian ini menggunakan *training* dan *testing test* sebanyak 5% dari keseluruhan *dataset*. Pada *dataset training* memiliki 1.048.575 baris data. Sedangkan, *dataset testing* memiliki 733.705 baris data. *Dataset* BoT-IoT memiliki 3 label, yaitu *attack*, *category*, dan *subcategory*.

Pada penelitian ini hanya menggunakan 10 fitur terbaik yang ditetapkan oleh UNSW Canberra. Fitur dan deskripsinya dapat dilihat pada Tabel 1.

Tabel 1 Fitur Terbaik dan Deskripsinya

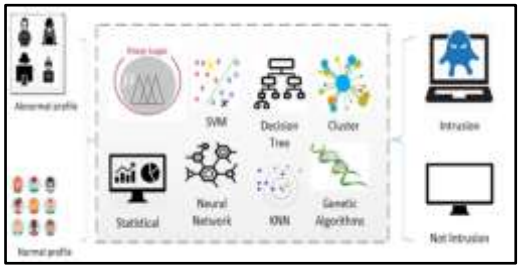
Fitur	Deskripsi
seq	Argus *sequence number* (Fitur ini diperoleh sebagai .argus format *file* dari *file* .pcap menggunakan Argus *tool*)
stddev	Standar deviasi dari *aggregated records*
min	Durasi minimum dari *aggregated records*
max	Durasi maksimum dari *aggregated records*
state_number	Representasi numerik dari status fitur
mean	Rata-rata *aggregated records*
N_IN_Conn_P_SrcIP	Jumlah koneksi masuk per IP sumber
N_IN_Conn_P_DstIP	Jumlah koneksi masuk per IP tujuan
drate	Paket tujuan-ke-sumber per detik
srate	Paket sumber-ke-tujuan per detik

3.2 *INTRUSION DETECTION SYSTEM*

Intrusion Detection System (IDS) merupakan proses pemantauan peristiwa yang terjadi pada sistem komputer atau jaringan dan menganalisisnya untuk menentukan apakah termasuk dalam keadaan normal atau intrusi. Model proses IDS memiliki 3 fungsi dasar, yaitu: pertama, pengambilan data dari berbagai tingkatan sistem seperti jaringan, *host*, dan aplikasi [2].

Secara umum, IDS dibagi menjadi dua kelompok yaitu *Signature-based intrusion detection systems* (SIDS) dan *Anomaly-based Intrusion Detection System* (AIDS). SIDS atau dikenal juga sebagai *misuse detection* didasarkan pada teknik pencocokan pola untuk menemukan serangan. Pada SIDS, metode pencocokan digunakan untuk menemukan intrusi sebelumnya. Dengan kata lain, ketika *signature* penyusup cocok dengan *signature* penyusup sebelumnya yang sudah ada di *database*, sinyal alarm akan dipicu. SIDS memiliki keterbatasan karena tidak dapat mendeteksi intrusi dengan penyusup baru (*zero-day attacks*) yang tidak ada dalam *database*. AIDS dapat mengatasi keterbatasan SIDS. Dalam AIDS, model normal dari

perilaku sistem komputer dibuat menggunakan *machine learning*, *statistical-based* atau metode *knowledge-based* [4]. Secara umum, konsep kerja dari pendekatan AIDS berdasarkan *machine learning* dapat dilihat pada Gambar 2.



Gambar 2 Konsep kerja dari pendekatan AIDS berdasarkan *machine learning*

AIDS bekerja dengan bantuan algoritma *machine learning* untuk mendeteksi intrusi. Pada penelitian ini akan menggunakan pendekatan AIDS berdasarkan algoritma klasifikasi dalam *machine learning*, yaitu *K-Nearest Neighbors*, *Random Forest*, *Gaussian Naive Bayes*. Selama *development*, AIDS terdiri dari dua fase: fase *training* dan fase *testing*. Pada fase *training*, profil lalu lintas normal digunakan untuk mempelajari model perilaku normal, dan kemudian pada fase *testing* kumpulan data baru digunakan untuk menetapkan kapasitas sistem untuk menggeneralisasi intrusi yang sebelumnya tidak terlihat [4].

3.3 *PREPROCESSING*

Preprocessing merupakan proses mengubah bentuk data tidak terstruktur menjadi data yang terstruktur. Tujuan dilakukannya *preprocessing* adalah memilih setiap kata dari dokumen dan merubahnya menjadi kata dasar yang memiliki arti sempit [2]. Pada penelitian ini dilakukan *data preprocessing* berupa label *encoding* untuk mengubah setiap nilai dalam kolom menjadi angka yang berurutan, kemudian melakukan *split* dan *scaling* pada data *training* dan *testing*.

3.4 *K-NEAREST NEIGHBORS*

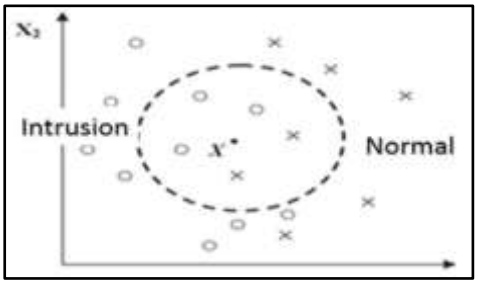
K-Nearest Neighbors (KNN) adalah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Metode ini bertujuan untuk mengklasifikasikan objek baru berdasarkan atribut dan *training sample*. Untuk pemilihan atribut terdiri dari n *neighbors* (biasa disebut k). Parameter k pada *testing* ditentukan berdasarkan nilai k optimum pada saat *training* yang diperoleh dengan mencoba-coba [5]. Cara kerja dari KNN yaitu dengan menghitung jarak antara objek dengan semua tetangganya, kemudian diurutkan dan dipilih k tetangga terdekat. Setelah itu, lihat semua kategori atau label dari k tetangga terdekat tersebut, kategori mayoritas akan menjadi prediksi kategori untuk objek tersebut.

Jarak atau *distance* dapat dihitung dengan beberapa cara, salah satunya menggunakan *euclidean distance*. Rumus dari *euclidean distance* dapat dilihat pada Persamaan (1).

$$D(a,b) = \sqrt{\sum_{k=1}^{d} (a_k - b_k)^2} \quad (1)$$

Berdasarkan Persamaan (1) dimana matriks D(a,b) merupakan jarak skalar dari dua vektor yaitu a (data latih) dan b (data uji). Pada penelitian ini menggunakan nilai n *neighbors*

atau k sebanyak 5. Gambaran dari kerja KNN menggunakan n *neighbors* 5 untuk pada IDS untuk mendeteksi intrusi dapat dilihat pada Gambar 3.



Gambar 3 Contoh Klasifikasi dengan KNN untuk k=5

3.5 *RANDOM FOREST*

Pembelajar mesin *Random Forest*, adalah pembelajar meta; artinya terdiri dari banyak individu peserta didik (*tree*). *Random Forest* menggunakan beberapa klasifikasi pohon acak untuk memilih klasifikasi keseluruhan untuk *set input* yang diberikan. Secara umum di setiap individu machine learner suara diberikan bobot yang sama. Dalam karya Breiman selanjutnya, algoritma ini dimodifikasi untuk melakukan pemungutan suara tanpa pembobotan dan pembobotan. *Forest* memilih klasifikasi individu yang berisi suara terbanyak [9]. Perhitungan *Raw Importance* yang digunakan menghitung hitungan benar yang tidak tersentuh, jumlah klasifikasi yang benar menggunakan data *out-of-bag* sebagai *set* pengujiannya. Nilai- nilai atribut kemudian diubah secara acak dalam contoh *out-of-bag*. Kumpulan data baru ini kemudian diuji untuk klasifikasi yang benar. Rata-rata jumlah ini untuk semua pohon di hutan adalah skor kepentingan mentah untuk atribut tertentu. Perhitungan *raw importance* dapat dilihat pada Persamaan (2).

$$raw\_importance_{variable[m]} = \frac{untouched\_count - variable_m count}{number\_of\_trees} \quad (2)$$

Representasi visualisasi dari algoritma *random forest* untuk membuat prediksi dapat dilihat pada Gambar 4.

Gambar 4 Visualisasi Model *Random Forest* membuat Prediksi.

3.6 *GAUSSIAN NAIVE BAYES*

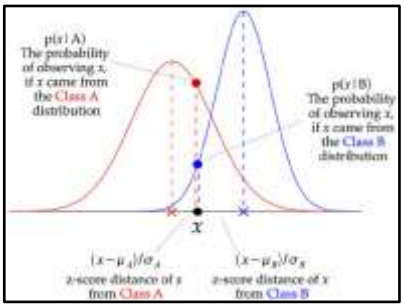
Gaussian Naïve Bayes (GNB) adalah algoritma pembelajaran mesin *supervised* yang menggunakan teorema *Bayes* sebagai kerangka kerja untuk mengklasifikasikan pengamatan ke

dalam salah satu set kelas yang telah ditentukan sebelumnya berdasarkan informasi yang diberikan oleh variabel prediktor. Pengklasifikasi GNB memperkirakan probabilitas bersyarat bahwa suatu pengamatan milik kelas tertentu yang diberikan nilai-nilai prediktor variabel dengan asumsi bahwa variabel prediktor adalah kelas-bersyarat independen, dan dengan demikian (*naive*) tidak memperhitungkan memperhitungkan kovarians di antara variabel-variabel prediktor [10]. Dengan demikian, probabilitas posterior bahwa pengamatan Y memiliki indeks kelas k yang diberikan nilai variabel prediktor X1, . . ., XP dimodelkan menurut Teorema *Bayes* sebagai:

$$\hat{P}(Y = k|X_1, \ldots, X_P) = \frac{\pi(Y = k)\prod_{j=1}^{P} P(X_j|Y = k)}{\sum_{k=1}^{K} \pi(Y = k)\prod_{j=1}^{P} P(X_j|Y = k)} \quad (3)$$

di mana π(Y = k) adalah probabilitas sebelumnya bahwa indeks kelas adalah k. Untuk setiap prediktor X1, . . ., Xp, algoritma akan memperkirakan Distribusi *Gaussian* terpisah untuk setiap kelas, dan observasi akan dilakukan ke kelas dengan probabilitas posterior maksimum yang didapat dari nilai prediktor.

Representasi visual dari algoritma *Gaussian Naive Bayes* untuk melakukan prediksi dapat dilihat pada Gambar 5.



Gambar 5 Visualisasi Model *Gaussian Naive Bayes*

3.7 *CONFUSION MATRIX*

Confusion matrix merupakan suatu metode untuk melakukan perhitungan akurasi, serta menganalisis seberapa baik baik *classifier* dapat mengenai tupel dari kelas yang berbeda. Misalkan terdapat dua kelas, maka akan diistilahkan sebagai *tupel positive* dan *tupel negativ*e. *True positive* mengacu pada *tuple positif* yang diberi label dengan tepat oleh *Classifier*, sementara *true negative* adalah *tupel positif* yang diberi label dengan tepat oleh *Classifier*. *False positive* adalah *tupel negatif* yang diberi label tidak tepat, *false negative* adalah *tupel positif* yang diberi label dengan tidak tepat [1]. *Confusion matrix* dapat dilihat pada Tabel 2.

Tabel 2 Model *Confusion Matrix*

		Predicted Class	
		Positive	*Negative*
Actual Class	*Positive*	*True positive* (TP)	*False negative* (FN)
	Negative	*False positive* (FP)	*True negative* (TN)

Dari *confusion matrix* dapat dihitung nilai akurasi, presisi, *recall*, dan F1-*scores*. Rumus *accuracy* dapat dilihat pada Persamaan (4), *precision* pada Persamaan (5), *recall* pada Persamaan (6), dan F1-*scores* pada Persamaan (7).

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

$$precision = \frac{TP}{TP+FP} \quad (5)$$

$$recall = \frac{TP}{TP+FN} \quad (6)$$

$$F1 - scores = \frac{2\ x\ precision\ x\ recall}{precision + recall} \quad (7)$$

Nilai terbaik F1-*Scores* adalah 1.0 dan nilai terburuknya adalah 0. Secara representasi, jika F1-*Scores* punya skor yang baik mengindikasikan bahwa model klasifikasi kita mempunyai *precision* dan *recall* yang baik.

4 HASIL DAN PEMBAHASAN

Penelitian ini memiliki tiga percobaan dalam pelatihan model dengan model yang berbeda-beda. Pertama menggunakan model *K-Nearest Neighbors (KNN)* dengan k = 5. Kedua menggunakan model *Random Forest* dengan *max_depth* = 5. Lalu yang terakhir menggunakan model *Gaussian Naive Bayes.* Klasifikasi dibagi menjadi jenis label seperti yang ditampilkan pada Tabel 3.

Tabel 2 Pengkodean 3 Jenis Kelas

	Kelas		
	Attack	*Category*	*SubCategory*
0	Normal	DDoS	UDP
1	Serangan / intrusi	DoS	TCP
2		Reconnaissance	Service Scan
3		Normal	OS Fingerprint
4		Theft	HTTP
5			Normal
6			Keylogging
7			Data Exfiltration

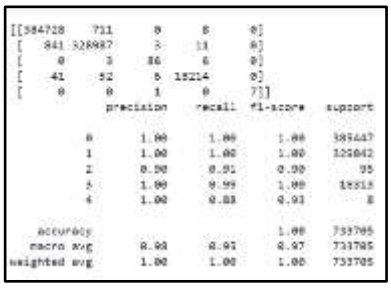
4. 1 *K-Nearest Neighbors (KNN)*

Telah dilakukan analisis validasi untuk *dataset training* terhadap *dataset* BoT-IoT dengan menggunakan model *K-Nearest Neighbors* pada 3 jenis label yang ada yaitu *attack*, *category*, dan *subcategory*. Sebelum melakukan validasi, dilakukan *training* menggunakan model KNN. *Training* pada model KNN membutuhkan waktu *running* 1140 detik. Hasil analisis validasi pada *training dataset* dengan label *attack* dapat dilihat pada Gambar 6, label *category* pada Gambar 7, dan label *sub category* pada Gambar 8.

```
[[    86      9]
 [    10 733600]]
              precision    recall  f1-score   support

           0       0.90      0.91      0.90        95
           1       1.00      1.00      1.00    733610

    accuracy                           1.00    733705
   macro avg       0.95      0.95      0.95    733705
weighted avg       1.00      1.00      1.00    733705
```

Gambar 6 Analisis Validasi pada *Training* untuk label "Attack" dengan KNN

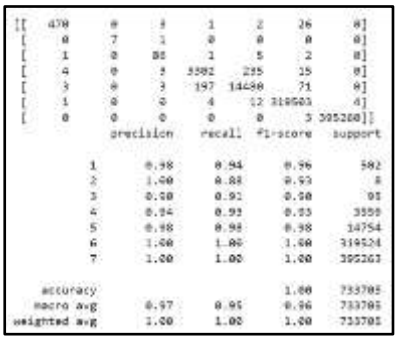
Setelah dilakukan analisis validasi pada *training dataset* untuk label "Attack" yang memiliki kelas 0 untuk aktifitas normal dan 1 untuk intrusi, didapatkan hasil berupa nilai *precision* sebesar 90%, *recall* sebesar 91%, *f1-score* sebesar 90% dan *accuracy* sebesar 100%.



```
[[584728    711      0      5      0]
 [   841 326987      3     11      0]
 [     0      3     86      6      0]
 [    41     52      5  15214      0]
 [     0      0      1      0      7]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    585447
           1       1.00      1.00      1.00    327842
           2       0.90      0.91      0.90        95
           3       1.00      0.99      1.00     15315
           4       1.00      0.88      0.93         8

    accuracy                           1.00    733705
   macro avg       0.98      0.91      0.97    733705
weighted avg       1.00      1.00      1.00    733705
```

Gambar 7 Analisis Validasi pada *Training* untuk label "*Category*" dengan KNN

Setelah dilakukan analisis validasi pada *training dataset* untuk label "*Category*", didapatkan hasil berupa nilai *precision* sebesar 100%, *recall* sebesar 100%, *f1-score* sebesar 100% dan *accuracy* sebesar 100%.

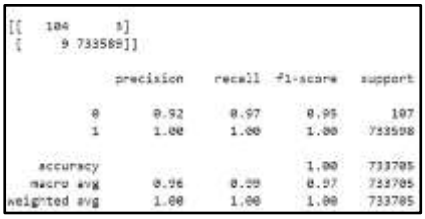


```
[[   470      0      5      1      2     26      0]
 [     0      7      1      0      0      0      0]
 [     1      0     86      1      5      2      0]
 [     4      0      5   3582    235     15      0]
 [     3      0      3    197  14438     71      0]
 [     1      0      0      4     12 318503      4]
 [     0      0      0      0      0      3 395268]]
              precision    recall  f1-score   support

           1       0.98      0.94      0.96       502
           2       1.00      0.88      0.93         8
           3       0.90      0.91      0.90        94
           4       0.94      0.93      0.93      3838
           5       0.98      0.98      0.98     14754
           6       1.00      1.00      1.00    319524
           7       1.00      1.00      1.00    395263

    accuracy                           1.00    733705
   macro avg       0.97      0.95      0.96    733705
weighted avg       1.00      1.00      1.00    733705
```

Gambar 8 Analisis Validasi pada *Training* untuk label "*Subcategory*" dengan KNN

Setelah dilakukan analisis validasi pada *training dataset* untuk label "*Subcategory*", didapatkan hasil berupa nilai *precision* sebesar 98%, *recall* sebesar 94%, *f1-score* sebesar 96% dan *accuracy* sebesar 100%.

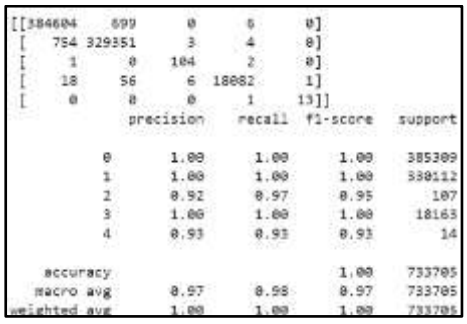
Selanjutnya dilakukan *testing* dengan model KNN. *Testing* pada model KNN membutuhkan waktu *running* 894.992s. Kemudian dilakukan *testing* pada *test set* menggunakan model KNN. Hasil analisis validasi pada *testing dataset* dengan label *attack* dapat dilihat pada Gambar 9, label *category* pada Gambar 10, dan label *subcategory* pada Gambar 11.



```
[[   184      1]
 [     9 733589]]
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       187
           1       1.00      1.00      1.00    733598

    accuracy                           1.00    733705
   macro avg       0.96      0.99      0.97    733705
weighted avg       1.00      1.00      1.00    733705
```

Gambar 9 Analisis Validasi pada *Testing* untuk label "Attack" dengan KNN

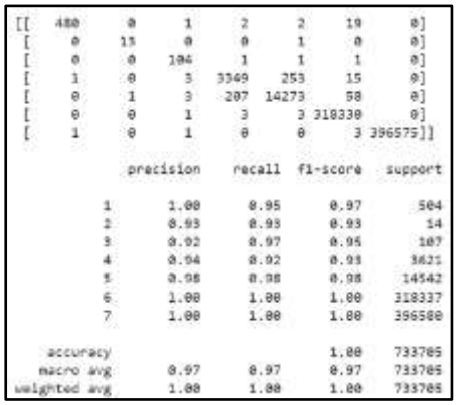
Setelah dilakukan analisis validasi pada *testing dataset* untuk label "Attack", didapatkan hasil berupa nilai *precision* sebesar 92%, *recall* sebesar 97%, *f1-score* sebesar 95% dan *accuracy* sebesar 100%.



```
[[384604    699      0      6      0]
 [   754 329351      3      4      0]
 [     1      0    104      2      0]
 [    18     56      6  18082      1]
 [     0      0      0      1     13]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    385309
           1       1.00      1.00      1.00    330112
           2       0.92      0.97      0.95       107
           3       1.00      1.00      1.00     18165
           4       0.93      0.93      0.93        14

    accuracy                           1.00    733705
   macro avg       0.97      0.98      0.97    733705
weighted avg       1.00      1.00      1.00    733705
```

Gambar 10 Analisis Validasi pada *Testing* untuk label "*Category*" dengan KNN

Setelah dilakukan analisis validasi pada *testing dataset* untuk label "*Category*", didapatkan hasil berupa nilai *precision* sebesar 100%, *recall* sebesar 100%, *f1-score* sebesar 100% dan *accuracy* sebesar 100%.



```
[[   480      0      1      2      2     19      0]
 [     0     15      0      0      1      0      0]
 [     0      0    104      1      1      1      0]
 [     1      0      3   3349    253     15      0]
 [     0      1      3    207  14273     58      0]
 [     0      0      1      3      3 318330      0]
 [     1      0      1      0      0      3 396575]]

              precision    recall  f1-score   support

           1       1.00      0.95      0.97       504
           2       0.93      0.93      0.93        14
           3       0.92      0.97      0.95       107
           4       0.94      0.92      0.93      3621
           5       0.98      0.98      0.98     14542
           6       1.00      1.00      1.00    318337
           7       1.00      1.00      1.00    396580

    accuracy                           1.00    733705
   macro avg       0.97      0.97      0.97    733705
weighted avg       1.00      1.00      1.00    733705
```

Gambar 11 Analisis Validasi pada *Testing* untuk label "*Subcategory*" dengan KNN

Setelah dilakukan analisis validasi pada *testing dataset* untuk label "*Subcategory*", didapatkan hasil berupa nilai *precision* sebesar 100%, *recall* sebesar 95%, *f1-score* sebesar 97% dan *accuracy* sebesar 100%.

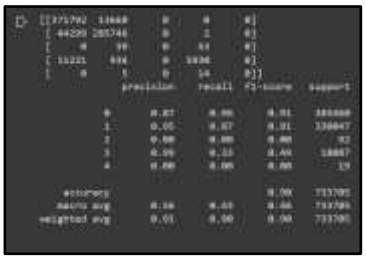
4. 2 *Random Forest*

Telah dilakukan analisis validasi untuk dataset *training* terhadap dataset BOT - IoT dengan menggunakan model *Random Forest* dengan 3 label yaitu *attack, Category,* dan *Sub Category*. Hasil analisis validasi pada *training dataset* dengan label *attack* dapat dilihat pada Gambar 12, dengan label *category* dapat dilihat pada Gambar 13, dan dengan label *Subcategory* dapat dilihat pada Gambar 14 dengan waktu *running* 587 detik. Hasilnya sebagai berikut:

```
[[     9     82]
 [     0 733613]]
              precision    recall  f1-score   support

           0       1.00      0.10      0.18        91
           1       1.00      1.00      1.00    733613

    accuracy                           1.00    733705
   macro avg       1.00      0.55      0.59    733705
weighted avg       1.00      1.00      1.00    733705
```

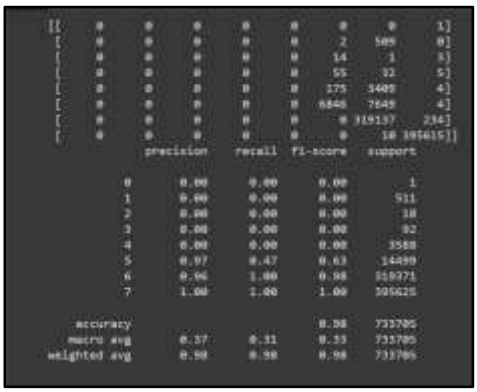
Gambar 12 Analisis Validasi pada Training untuk "Attack"

Setelah melakukan analisis validasi didapat *precision* sebesar 100%, *recall* sebesar 10%, *f1 - score* sebesar 18%, dan *accuracy* sebesar 100% untuk "Attack".



Gambar 13 Analisis Validasi pada Training untuk "*Category*"

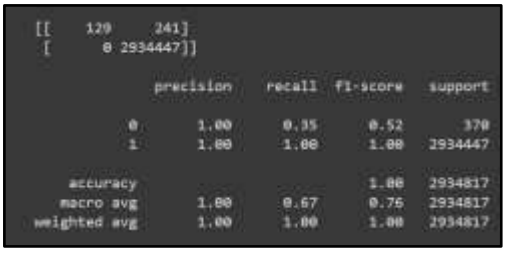
Setelah melakukan analisis validasi didapat *precision* sebesar 87%, *recall* sebesar 96%, *f1 - score* sebesar 91%, dan *accuracy* sebesar 90% untuk "*Category".*



Gambar 14 Analisis Validasi pada Training untuk "*Subcategory*"

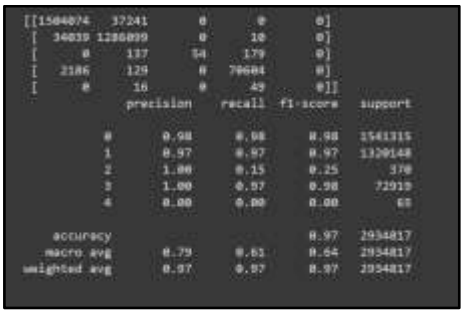
Setelah melakukan analisis validasi didapat *precision* sebesar 0%, *recall* sebesar 0%, *f1 - score* sebesar 0%, dan *accuracy* sebesar 98% untuk "*Subcategory".*

Setelah melakukan analisis validasi untuk dataset *training* terhadap dataset BOT - IoT, selanjutnya dilakukan *testing* pada *testing dataset*. Hasil *testing* pada *testing dataset* dengan label *attack* dapat dilihat pada Gambar 15, dengan label *category* dapat dilihat pada Gambar 16, dan dengan label *Subcategory* dapat dilihat pada Gambar 17 dengan waktu *running* 1584.145s. Hasilnya adalah sebagai berikut:



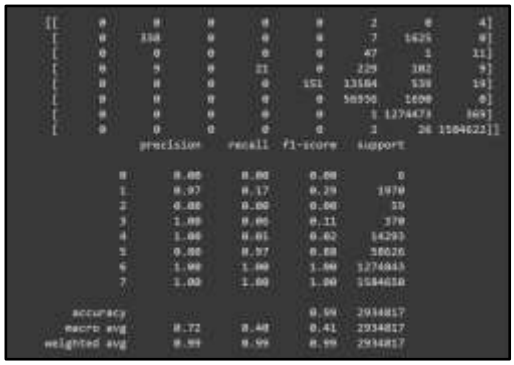
Gambar 15 *Testing* pada *Testing Dataset* untuk "Attack"

Setelah melakukan *Testing* pada *Testing Dataset* didapat *precision* sebesar 100%, *recall* sebesar 35%, *f1 - score* sebesar 52%, dan *accuracy* sebesar 100% untuk "Attack".



Gambar 16 *Testing* pada *Testing Dataset* untuk "*Category*"

Setelah melakukan *Testing* pada *Testing Dataset* didapat *precision* sebesar 98%, *recall* sebesar 98%, *f1 - score* sebesar 98%, dan *accuracy* sebesar 97% untuk "*Category*".

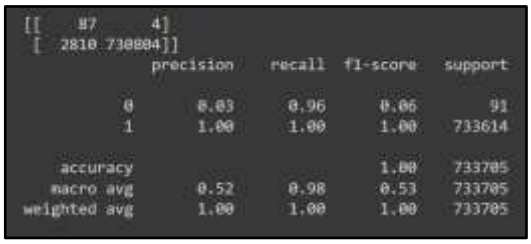


Gambar 17 *Testing* pada *Testing Dataset* untuk "*Subcategory*"

Setelah melakukan *Testing* pada *Testing Dataset* didapat *precision* sebesar 0%, *recall* sebesar 0%, *f1 - score* sebesar 0%, dan *accuracy* sebesar 99% untuk "*Subcategory*".

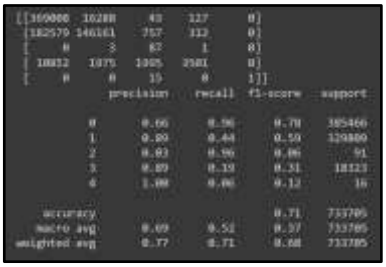
4. 3 *Gaussian Naive Bayes*

Telah dilakukan analisis validasi untuk dataset *training* terhadap dataset BOT - IoT dengan menggunakan model *Gaussian Naive Bayes* dengan tiga label yaitu *Attack, Category,* dan *Sub Category*. Waktu eksekusi selama 29,28 detik. Hasil analisis validasi pada *training dataset* dengan label *Attack* dapat dilihat pada Gambar 18, dengan label *Category* dapat dilihat pada Gambar 19, dan dengan label *Subcategory* pada Gambar 20.



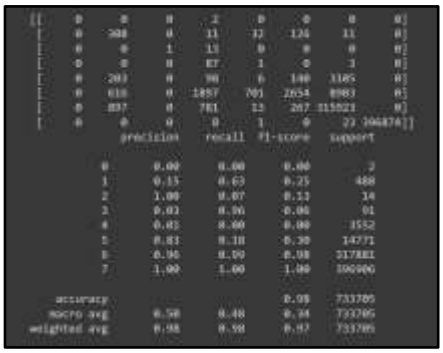
Gambar 18 Analisis Validasi pada "*Attack*" dengan Gaussian Naive Bayes

Setelah dilakukan analisis validasi pada *training dataset* untuk label "Attack", didapatkan hasil berupa nilai *precision* sebesar 3%, *recall* sebesar 96%, *f1-score* sebesar 6% dan *accuracy* sebesar 100%.



Gambar 19 Analisis Validasi pada *"Category"* dengan Gaussian Naive Bayes

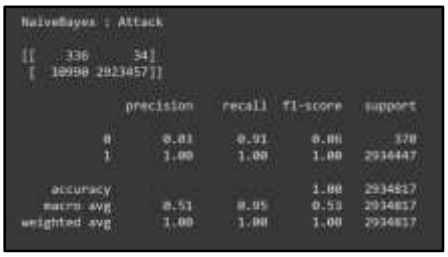
Setelah dilakukan analisis validasi pada *training dataset* untuk label "*Category*", didapatkan hasil berupa nilai *precision* sebesar 66%, *recall* sebesar 96%, *f1-score* sebesar 78% dan *accuracy* sebesar 71%.



Gambar 20 Analisis Validasi pada *"Subcategory"* dengan Gaussian Naive Bayes

Setelah dilakukan analisis validasi pada *training dataset* untuk label "*Subcategory*", didapatkan hasil berupa nilai *precision* sebesar 0%, *recall* sebesar 0%, *f1-score* sebesar 0% dan *accuracy* sebesar 98%.

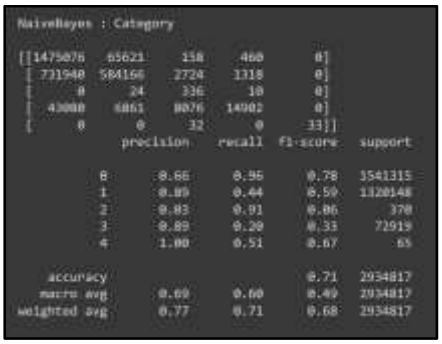
Setelah melakukan analisis validasi untuk dataset *training* terhadap dataset BOT - IoT, selanjutnya dilakukan *testing* pada *testing dataset*. Hasil *testing* pada *testing dataset* dengan label *attack* dapat dilihat pada Gambar 21, dengan label *category* dapat dilihat pada Gambar 22, dan dengan label *Subcategory* dapat dilihat pada Gambar 23 dengan waktu *running* 28,98 detik.



Gambar 21 *Testing* pada *Testing Dataset* dengan *Gaussian NB untuk "Attack"*

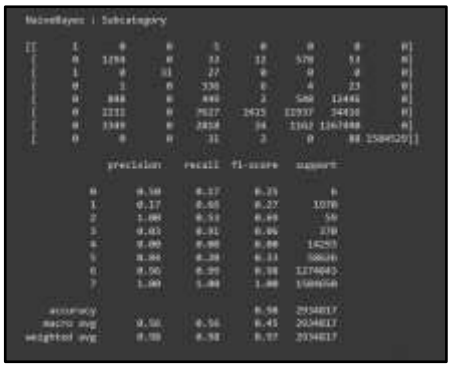
Setelah melakukan *testing* pada *Testing Dataset* dengan model Gaussian NB, didapat nilai *precision* sebesar 3%, *recall* sebesar 91%, *f1-score* sebesar 6%, dan *accuracy* sebesar 100% untuk "Attack".

ISSN: 2086-4930
e-ISSN: 2777-0648



Gambar 22 *Testing* pada *Testing Dataset* dengan *Gaussian NB untuk "Category"*

Setelah melakukan *testing* pada *Testing Dataset* dengan model Gaussian NB, didapat nilai *precision* sebesar 66%, *recall* sebesar 96%, *f1-score* sebesar 78%, dan *accuracy* sebesar 71% untuk "*Category*".

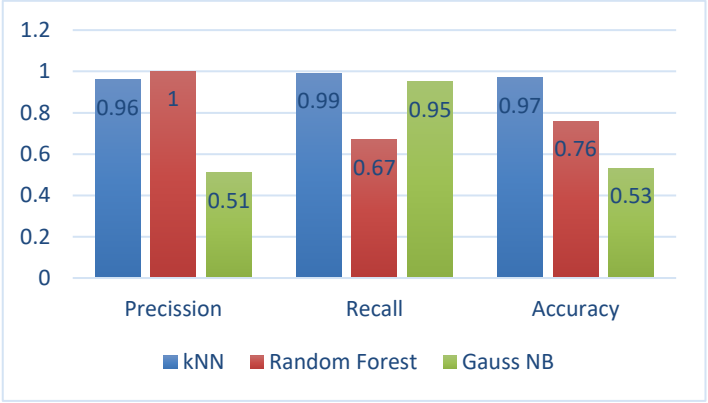


Gambar 23 *Testing* pada *Testing Dataset* dengan *Gaussian NB untuk "Subcategory"*

Setelah melakukan *testing* pada *Testing Dataset* dengan model Gaussian NB, didapat nilai *precision* sebesar 56%, *recall* sebesar 17%, *f1-score* sebesar 25%, dan *accuracy* sebesar 98% untuk "*Subcategory*".

4. 4 Hasil Analisis Semua Model

Hasil penelitian yang telah didapatkan bahwa model terbaik adalah *K-Nearest Neighbors (KNN)* karena memiliki nilai *precision, recall, f1 - Score,* dan *Accuracy* terbaik dibandingkan 2 model lainnya. *K-Nearest Neighbors (KNN)* juga memenuhi standar untuk model dikatakan baik/cocok diterapkan pada *dataset* BoT-IoT karena menghasilkan nilai *recall*, *precision,* dan *f1-score* yang tinggi. Seperti yang terlihat pada Gambar 24



Gambar 24 Perbandingan performa algoritma pada "Attack"

5 KESIMPULAN

Berdasarkan hasil dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa metode KNN cocok diterapkan pada *dataset* BoT-IoT karena menghasilkan nilai *recall*, *precision,* dan *f1-score* yang tinggi. Sedangkan, metode *Random Forest* dan *Gaussian Naive Bayes* kurang cocok diterapkan pada *dataset* BoT-IoT karena pa da *Random Forest* menghasilkan nilai *recall* dan *f1-score* yang cukup rendah, dan pada *Gaussian Naive Bayes* menghasilkan nilai *precision* dan *f1-score* yang sangat rendah. KNN memang memiliki nilai evaluasi untuk prediksi yang besar, namun memiliki kelemahan yaitu memerlukan waktu yang cukup lama untuk melakukan proses *training* dan *testing* dibandingkan menggunakan metode *Gaussian Naive Bayes*.

DAFTAR PUSTAKA

[1] E. Özer, M. İskefiyeli, and J. Azimjonov, "Toward lightweight intrusion detection systems using the optimal and efficient feature pairs of the Bot-IoT 2018 dataset," *Int. J. Distrib. Sens. Networks*, vol. 17, no. 10, pp. 1–20, 2021, doi: 10.1177/15501477211052202.

[2] I. N. T. Wirawan and I. Eksistyanto, "Penerapan Naive Bayes Pada Intrusion Detection System Dengan Diskritisasi Variabel," *JUTI J. Ilm. Teknol. Inf.*, vol. 13, no. 2, p. 182, 2015, doi: 10.12962/j24068535.v13i2.a487.

[3] N. Koroniotis and N. Moustafa, "The Bot-IoT Dataset," 2021. https://research.unsw.edu.au/projects/bot-iot-dataset (accessed Oct. 11, 2022).

[4] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, 2019, doi: 10.1186/s42400-019-0038-7.

[5] N. Suwaryo, I. Nawangsih, and S. Rejeki, "Deteksi Serangan pada Intrusion Detection System (IDS) untuk Klasifikasi Serangan dengan Algoritma Naïve Bayes, C.45 dan K-NN dalam Meminimalisasi Resiko Terhadap Pengguna," *J. Sist. Inf. Univ. Suryadarma*, vol. 8, no. 2, pp. 171–180, 2021, doi: 10.35968/jsi.v8i2.732.

[6] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Futur. Gener. Comput. Syst.*, vol. 100, pp. 779–796, 2019, doi: 10.1016/j.future.2019.05.041.

[7] S. Pokhrel, R. Abbas, and B. Aryal, "IoT Security: Botnet detection in IoT using Machine learning," pp. 1–11, 2021, [Online]. Available: http://arxiv.org/abs/2104.02231

[8] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city," *Futur. Gener. Comput. Syst.*, vol. 107, pp. 433–442, 2020, doi: 10.1016/j.future.2020.02.017.

[9] F. Livingston, "Implementation of Breiman's Random Forest Machine Learning Algorithm," *Mach. Learn. J. Pap.*, pp. 1–13, 2005

[10] J. C. Griffis, J. B. Allendorfer, and J. P. Szaflarski, "Voxel-based Gaussian naïve Bayes classification of ischemic stroke lesions in individual T1-weighted MRI scans," *J Neurosci Methods*, vol. 257, pp. 97–108, Jan. 2016, doi: 10.1016/j.jneumeth.2015.09.019.