BAB III
METODE PENELITIAN

3.1 Objek Penelitian

Objek penelitian ini adalah dataset serangan *DDoS* yang digunakan untuk melatih dan menguji model machine learning. Dataset yang dipilih adalah CICIDS-2017 yang dikembangkan oleh *Canadian Institute for Cybersecurity* Objek penelitian ini adalah data lalu lintas jaringan yang mengandung serangan *Distributed Denial of Service* (*DDoS*). Dataset yang digunakan dalam penelitian ini diambil dari sumber yang telah digunakan dalam berbagai penelitian keamanan siber, seperti CICIDS-2017 dan dataset lain yang relevan. Dataset ini mencakup berbagai jenis serangan DDoS dengan pola lalu lintas yang bervariasi, termasuk serangan berbasis volumetrik, protokol, dan aplikasi.

Pemilihan dataset ini didasarkan pada karakteristiknya yang mencerminkan kondisi nyata dalam lingkungan jaringan modern. Dataset ini memiliki fitur yang mencakup parameter lalu lintas jaringan seperti jumlah paket, ukuran paket, protokol yang digunakan, serta pola komunikasi antara sumber dan tujuan. Data ini kemudian digunakan untuk melatih dan menguji berbagai algoritma *Machine Learning* yang dibandingkan dalam penelitian ini. Selain dataset, penelitian ini juga menggunakan perangkat lunak dan alat analisis data yang relevan untuk melakukan *preprocessing*, pelatihan model, dan evaluasi performa algoritma.

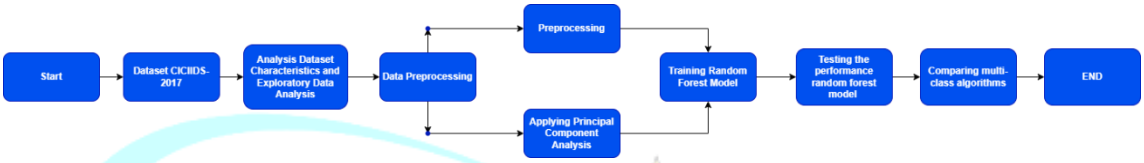
Dengan pemilihan objek penelitian yang sesuai, penelitian ini bertujuan untuk mengembangkan dan menguji efektivitas berbagai algoritma *Machine Learning* dalam mendeteksi serangan *DDoS* serta mengevaluasi performa masing-masing metode berdasarkan akurasi, presisi, *recall*, dan efisiensi komputasi.

3.2 Jenis Penelitian

Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimen. Tujuan dari metode ini adalah untuk menguji efektivitas algoritma machine learning khususnya Random Forest dalam mendeteksi dan mengklasifikasikan berbagai jenis serangan jaringan berdasarkan dataset CIC-IDS2017. Dengan pendekatan ini, model dievaluasi secara objektif menggunakan metrik evaluasi seperti accuracy, precision, recall, dan f1-score.

3.3 Prosedur Penelitian

Prosedur penelitian ini disusun secara sistematis untuk memastikan bahwa setiap langkah dapat dilakukan dengan efektif dan menghasilkan data yang relevan dengan tujuan penelitian. Berikut adalah tahapan-tahapan dalam prosedur penelitian :

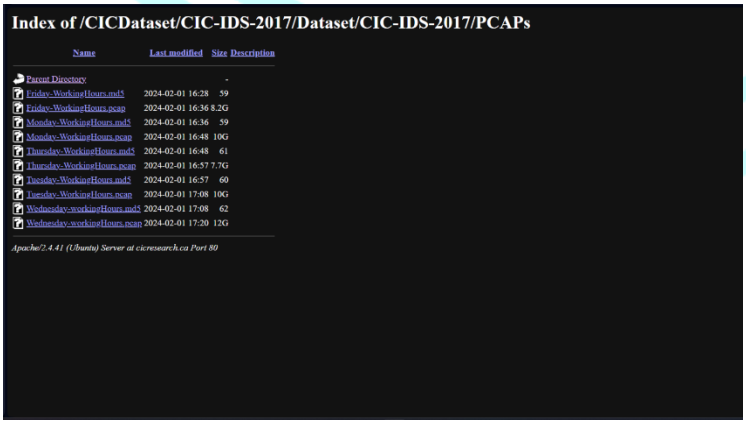


Gambar 3. 1 Kerangka Penelitian

Prosedur penelitian ini dirancang untuk membandingkan performa lima algoritma machine learning dalam deteksi *DDoS*. Tahapan penelitian dijelaskan sebagai berikut :

3.3.1 Langkah 1: Pengumpulan Data

Dataset yang digunakan pada penelitian ini terdiri dari 2.502.798 data dengan 79 fitur yang mewakili berbagai karakteristik lalu lintas jaringan. Atribut ini mencakup karakteristik seperti *Destination Port, Flow Duration, Total Fwd Packets, Flow Bytes/s, Average Packet Size, Idle Mean, dan Label*. Data ini dibagi menjadi dua kelas: *BENIGN*, yang mewakili lalu lintas jaringan normal, dan *DDoS*, yang mewakili lalu lintas jaringan yang mewakili *DDoS*.



Gambar 3. 2 Dataset CICIDS-2017

3.3.2 Langkah 2: Analysis Dataset Characteristics and Exploratory Data Analysis

Pada tahap ini dilakukan analisis awal terhadap dataset CIC-IDS2017 untuk memahami struktur, pola, dan distribusi data yang digunakan dalam proses pelatihan model deteksi serangan *DDoS*. Dataset terdiri dari lebih dari dua juta sampel lalu lintas jaringan, yang mencakup berbagai fitur seperti *Flow Duration*, *Total Fwd Packets*, *Flow Bytes/s*, hingga *Packet Length Mean*. Fitur-fitur ini digunakan untuk mengidentifikasi perbedaan antara lalu lintas normal dan serangan.

	count	mean	std	min	25%	50%	75%	max
Destination Port	2830743.0	8.071483e+03	1.828363e+04	0.000000e+00	53.000000	80.000000	4.430000e+02	6.553500e+04
Flow Duration	2830743.0	1.478566e+07	3.365374e+07	-1.300000e+01	155.000000	31316.000000	3.204828e+06	1.200000e+08
Total Fwd Packets	2830743.0	9.361160e+00	7.496728e+02	1.000000e+00	2.000000	2.000000	5.000000e+00	2.197590e+05
Total Backward Packets	2830743.0	1.039377e+01	9.973883e+02	0.000000e+00	1.000000	2.000000	4.000000e+00	2.919220e+05
Total Length of Fwd Packets	2830743.0	5.493024e+02	9.993589e+03	0.000000e+00	12.000000	62.000000	1.870000e+02	1.290000e+07
Total Length of Bwd Packets	2830743.0	1.616264e+04	2.263088e+06	0.000000e+00	0.000000	123.000000	4.820000e+02	6.554530e+08
Fwd Packet Length Max	2830743.0	2.075999e+02	7.171848e+02	0.000000e+00	6.000000	37.000000	8.100000e+01	2.482000e+04
Fwd Packet Length Min	2830743.0	1.871366e+01	6.033935e+01	0.000000e+00	0.000000	2.000000	3.600000e+01	2.325000e+03
Fwd Packet Length Mean	2830743.0	5.820194e+01	1.860912e+02	0.000000e+00	6.000000	34.000000	5.000000e+01	5.940857e+03
Fwd Packet Length Std	2830743.0	6.891013e+01	2.811871e+02	0.000000e+00	0.000000	0.000000	2.616295e+01	7.125597e+03
Bwd Packet Length Max	2830743.0	8.708495e+02	1.946367e+03	0.000000e+00	0.000000	79.000000	2.800000e+02	1.953000e+04
Bwd Packet Length Min	2830743.0	4.104958e+01	6.886260e+01	0.000000e+00	0.000000	0.000000	7.700000e+01	2.896000e+03
Bwd Packet Length Mean	2830743.0	3.059493e+02	6.052568e+02	0.000000e+00	0.000000	72.000000	1.810000e+02	5.800500e+03
Bwd Packet Length Std	2830743.0	3.353257e+02	8.396932e+02	0.000000e+00	0.000000	0.000000	7.794054e+01	8.194660e+03
Flow Bytes/s	2829385.0	inf	NaN	-2.610000e+08	119.319731	4595.549126	1.666667e+05	inf
Flow Packets/s	2830743.0	inf	NaN	-2.000000e+06	3.446226	110.668437	2.325581e+04	inf
Flow IAT Mean	2830743.0	1.298449e+06	4.507944e+06	-1.300000e+01	63.666667	11438.842110	3.374266e+05	1.200000e+08
Flow IAT Std	2830743.0	2.919271e+06	8.045870e+06	0.000000e+00	0.000000	137.178716	6.912663e+05	8.480026e+07
Flow IAT Max	2830743.0	9.182475e+06	2.445954e+07	-1.300000e+01	123.000000	30865.000000	2.440145e+06	1.200000e+08
Flow IAT Min	2830743.0	1.623796e+05	2.950282e+06	-1.400000e+01	3.000000	4.000000	6.400000e+01	1.200000e+08
Fwd IAT Total	2830743.0	1.448296e+07	3.357581e+07	0.000000e+00	0.000000	43.000000	1.242844e+06	1.200000e+08
Fwd IAT Mean	2830743.0	2.610193e+06	9.525722e+06	0.000000e+00	0.000000	26.000000	2.063064e+05	1.200000e+08
Fwd IAT Std	2830743.0	3.266957e+06	9.639055e+06	0.000000e+00	0.000000	0.000000	6.598982e+04	8.460293e+07
Fwd IAT Max	2830743.0	9.042939e+06	2.452916e+07	0.000000e+00	0.000000	43.000000	9.310060e+05	1.200000e+08

Gambar 3. 3 Fitur dataset Cicids-2017

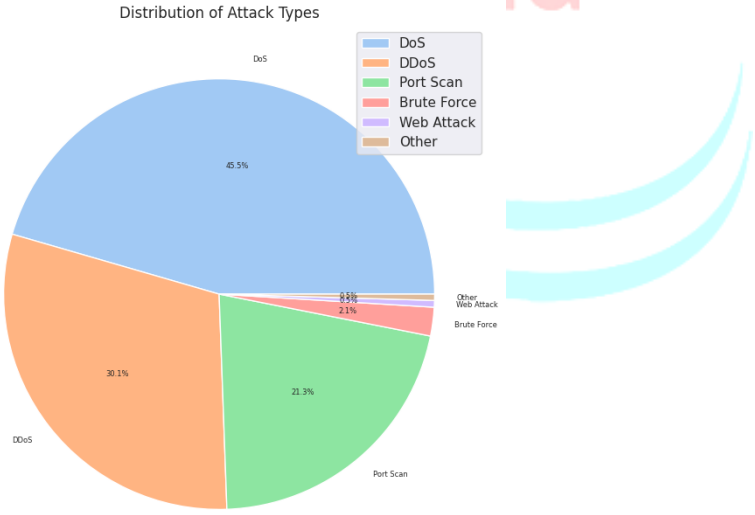
Proses *EDA* diawali dengan pembersihan data, yaitu menghapus nilai duplikat, memperbaiki format kolom, serta menangani nilai yang hilang atau tidak terdefinisi (*NaN* dan *infinite*). Nilai tak hingga diganti dengan NaN, lalu diisi menggunakan nilai median. Selain itu, dilakukan juga penggabungan label serangan ke dalam kategori utama seperti *DoS, DDoS, Port Scan, Brute Force, Web Attack*, dan *BENIGN*, yang dituangkan dalam kolom baru bernama "*Attack Type*".

Attack Type	count
BENIGN	2095057
DoS	193745
DDoS	128014
Port Scan	90694
Brute Force	9150
Web Attack	2143
Bot	1948
Infiltration	36
Heartbleed	11

dtype: int64

Gambar 3. 4 Pemberian Label *Attack Type*

Untuk memahami distribusi label dan mendeteksi ketidakseimbangan data, digunakan visualisasi seperti *pie chart* dan *bar chart*. Selain itu, dilakukan analisis korelasi antar fitur numerik dengan heatmap, guna mengidentifikasi fitur-fitur yang redundan atau berkorelasi tinggi. Analisis ini berguna untuk menyusun strategi seleksi fitur selanjutnya agar pelatihan model menjadi lebih efisien. Hasil *EDA* ini menjadi dasar yang kuat dalam pengambilan keputusan pada proses rekayasa fitur dan pemilihan model *machine learning*.



Gambar 3. 5 Distribusi Tipe Serangan

3.3.3 Langkah 3: Data Preprocessing

Tahap pra-pemrosesan data dilakukan untuk memastikan bahwa dataset dalam kondisi bersih, konsisten, dan sesuai untuk digunakan dalam pelatihan model klasifikasi. Dataset CIC-IDS2017 yang digunakan dalam penelitian ini memiliki dimensi besar dan kompleksitas tinggi, sehingga diperlukan sejumlah langkah sistematis dalam proses pembersihannya. Adapun tahapan-tahapan pra-pemrosesan data yang dilakukan antara lain :

a. Menghapus Duplikasi Data

   Dataset awal ditemukan mengandung banyak baris duplikat yang identik. Baris-baris tersebut dihapus untuk mencegah pembelajaran berulang oleh model dan meminimalkan risiko overfitting. Penghapusan ini juga membantu mempercepat proses pelatihan dan menghemat memori komputasi.

b. Menangani Nilai Tidak Valid (*NaN* dan *Inf*)

   Beberapa fitur seperti *Flow Bytes/s* dan *Flow Packets/s* mengandung nilai tidak valid seperti *NaN* dan *inf* yang dapat menyebabkan kegagalan saat pelatihan model. Nilai inf diubah menjadi *NaN*, dan seluruh nilai *NaN* diisi menggunakan median dari masing-masing kolom untuk menjaga distribusi data agar tetap representatif tanpa terpengaruh oleh *outlier*.

c. Mengelompokkan Label Serangan (*Attack Mapping*)

   Kolom Label pada dataset CIC-IDS2017 memiliki kategori yang sangat spesifik, seperti DDoS, DoS Hulk, PortScan, dan sebagainya. Untuk menyederhanakan klasifikasi, label-label tersebut dikelompokkan ulang ke dalam kategori utama seperti *DoS, DDoS, Brute Force, Port Scan, Web Attack, dan BENIGN*, yang disimpan dalam kolom baru bernama *Attack Type*.

d. *Encoding Label*

   Karena algoritma machine learning seperti *Random Forest* tidak dapat memproses data dalam bentuk teks, maka kolom target *Attack Type* dikonversi menjadi bentuk numerik menggunakan

label encoding, di mana setiap jenis serangan diberi nilai numerik unik, misalnya *BENIGN* = 0, *DDoS* = 1, dan seterusnya.

e. Pemeriksaan Struktur Data

Seluruh fitur diperiksa kembali untuk memastikan tidak ada data kosong, tipe data sesuai, dan siap diproses lebih lanjut dalam tahapan *feature engineering* dan pelatihan model.

Gambar 3. 6 Proses data *preprocessing*

3.3.4 Langkah 4: *Applying Principal Component Analysis*

Principal Component Analysis (*PCA*) merupakan teknik reduksi dimensi yang digunakan untuk menyederhanakan jumlah fitur dalam dataset tanpa kehilangan informasi penting. Dalam konteks penelitian ini, *PCA* digunakan untuk mengurangi kompleksitas data yang memiliki ratusan atribut numerik, sehingga model *Random Forest* dapat bekerja lebih efisien tanpa mengorbankan performa klasifikasi.

PCA bekerja dengan cara mengubah fitur asli menjadi himpunan baru yang disebut komponen utama (*principal components*), yang merupakan kombinasi linier dari fitur asli namun tidak berkorelasi satu sama lain. Komponen-komponen utama ini kemudian dipilih berdasarkan kontribusinya terhadap variansi total data.

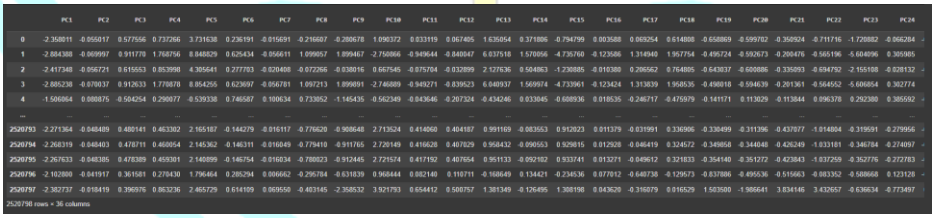
Dalam penelitian ini, digunakan metode Incremental *PCA*, yang merupakan versi *PCA* yang dioptimalkan untuk menangani dataset berukuran besar secara bertahap (*batch-wise*), sehingga lebih hemat memori dan komputasi. Proses ini dilakukan dengan menetapkan sejumlah

komponen utama (*n_components*) yang akan digunakan berdasarkan evaluasi terhadap proporsi variansi kumulatif. Pemilihan jumlah komponen mempertimbangkan agar setidaknya lebih dari 95% informasi data asli tetap dipertahankan.

Penerapan *PCA* membawa beberapa manfaat, yaitu:

a. Mengurangi risiko *overfitting* dengan menghilangkan fitur-fitur yang redundan atau tidak penting.
b. Meningkatkan efisiensi pelatihan model, karena jumlah fitur yang diproses menjadi lebih sedikit.
c. Meningkatkan generalisasi model, dengan menyederhanakan representasi data.

Secara keseluruhan, penggunaan *PCA* sebagai bagian dari tahapan feature engineering membantu menghasilkan model deteksi serangan jaringan yang lebih cepat, ringan, dan tetap akurat saat diterapkan dalam pengujian.



Gambar 3. 7 Penerapan *principal component analysis*

3.3.5 Langkah 5: *Training Random Forest Model*

Setelah melalui tahap pra-pemrosesan dan rekayasa fitur, proses selanjutnya adalah pelatihan model klasifikasi menggunakan algoritma Random Forest. Random Forest merupakan algoritma berbasis ensemble learning yang terdiri dari sejumlah pohon keputusan (*decision trees*) yang dibangun secara acak. Keputusan akhir model diambil berdasarkan voting mayoritas dari seluruh pohon, sehingga membuat model ini tahan terhadap overfitting dan efektif untuk klasifikasi multikelas seperti dalam deteksi serangan jaringan. Pelatihan model dilakukan dalam dua konfigurasi, yaitu Model 1 dan Model 2. Model 1 merupakan baseline model yang dilatih menggunakan parameter default dari pustaka *scikit-learn*. Model 2

merupakan model yang telah dioptimasi, salah satunya dengan menggunakan teknik reduksi dimensi (*PCA*) serta penyesuaian parameter seperti jumlah estimator (*n_estimators*), kedalaman pohon (*max_depth*), dan nilai acak (random_state) untuk meningkatkan performa klasifikasi.

Proses pelatihan dilakukan dengan membagi data menjadi dua bagian, yaitu data pelatihan (*training set*) dan data pengujian (*testing set*) menggunakan skema train-test split sebesar 80:20. Data pelatihan digunakan untuk membangun model, sedangkan data pengujian digunakan untuk mengevaluasi kinerjanya. Model dilatih menggunakan fitur numerik sebagai input dan kolom *Attack Type* yang telah diencoding sebagai target klasifikasi. Setelah pelatihan, model disimpan dan digunakan dalam tahap evaluasi untuk menghitung metrik performa seperti *accuracy*, precision, *recall*, dan *f1-score*. Dengan pendekatan ini, *Random Forest* menjadi algoritma utama dalam penelitian karena kemampuannya yang baik dalam menangani data besar, multiklasifikasi, dan fitur-fitur yang saling berkorelasi tinggi, seperti yang umum dijumpai dalam lalu lintas jaringan.

```
from sklearn.ensemble import RandomForestClassifier

rf1 = RandomForestClassifier(n_estimators = 10, max_depth = 6, max_features = None, random_state = 0)
rf1.fit(X_train, y_train)

cv_rf1 = cross_val_score(rf1, X_train, y_train, cv = 5)
print('Random Forest Model 1')
print(f'\nCross-validation scores:', ', '.join(map(str, cv_rf1)))
print(f'\nMean cross-validation score: {cv_rf1.mean():.2f}')
```

```
Random Forest Model 1

Cross-validation scores: 0.9664444444444444, 0.9651111111111111, 0.9648888888888889, 0.9653333333333334, 0.9671111111111111

Mean cross-validation score: 0.97
```

Gambar 3. 8 *Training model 1 random forest*

```
rf2 = RandomForestClassifier(n_estimators = 15, max_depth = 8, max_features = 20, random_state = 0)
rf2.fit(X_train, y_train)

cv_rf2 = cross_val_score(rf2, X_train, y_train, cv = 5)
print('Random Forest Model 2')
print(f'\nCross-validation scores:', ', '.join(map(str, cv_rf2)))
print(f'\nMean cross-validation score: {cv_rf2.mean():.2f}')
```

```
Random Forest Model 2

Cross-validation scores: 0.986, 0.984, 0.9815555555555555, 0.9784444444444444, 0.984

Mean cross-validation score: 0.98
```

Gambar 3. 9 *Training model 2 random forest*

3.3.6 Langkah 6: *Testing The Performance Random Forest Model*

Setelah proses pelatihan selesai, langkah berikutnya adalah menguji performa model Random Forest dengan menggunakan data uji (*testing data*) yang telah dipisahkan sebelumnya. Tujuan dari pengujian ini adalah untuk mengukur kemampuan generalisasi model dalam mengklasifikasikan jenis serangan pada data baru yang belum pernah dilihat selama proses pelatihan.

Pengujian dilakukan dengan memberikan input data uji kepada model yang telah dilatih, dan membandingkan hasil prediksi model dengan label sebenarnya. Untuk menilai performa model secara objektif, digunakan beberapa metrik evaluasi klasifikasi, yaitu :

a. *Accuracy*: Mengukur seberapa banyak prediksi model yang benar dibandingkan dengan total prediksi.
b. *Precision*: Mengukur ketepatan model dalam mengidentifikasi kelas tertentu, terutama penting dalam menghindari *false positive*.
c. *Recall*: Mengukur kemampuan model dalam mendeteksi seluruh instance dari kelas tertentu (*true positive*).
d. *F1-Score*: Rata-rata harmonis dari *precision* dan *recall*, digunakan untuk menilai keseimbangan performa model.

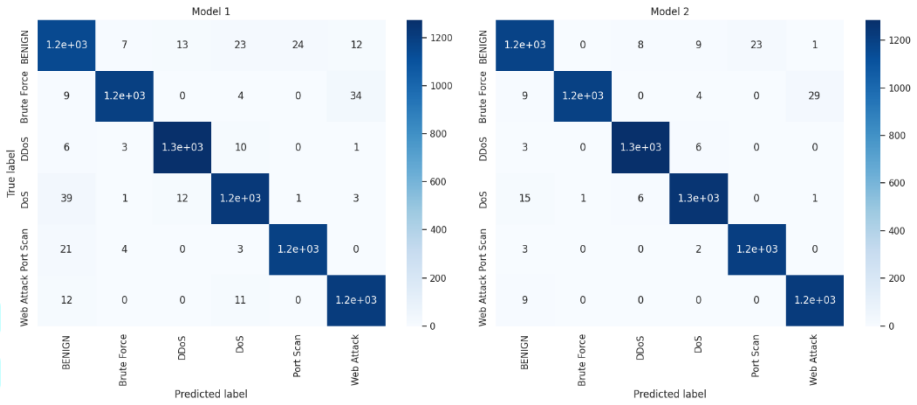
Selain itu, digunakan juga *confusion matrix* untuk melihat distribusi prediksi per kelas secara lebih rinci, serta mengidentifikasi di mana model melakukan kesalahan klasifikasi. Pengujian dilakukan terhadap dua versi model:

a. Model 1 (tanpa optimasi), dan
b. Model 2 (dengan *PCA* dan *tuning parameter*).

Hasil pengujian menunjukkan adanya peningkatan performa pada Model 2, yang ditandai dengan nilai evaluasi yang lebih tinggi dan distribusi prediksi yang lebih tepat pada *confusion matrix*.

Secara keseluruhan, tahap ini memastikan bahwa model yang dibangun tidak hanya bekerja dengan baik pada data pelatihan, tetapi juga memiliki

kinerja yang solid dan dapat diandalkan saat diterapkan pada data baru di dunia nyata.



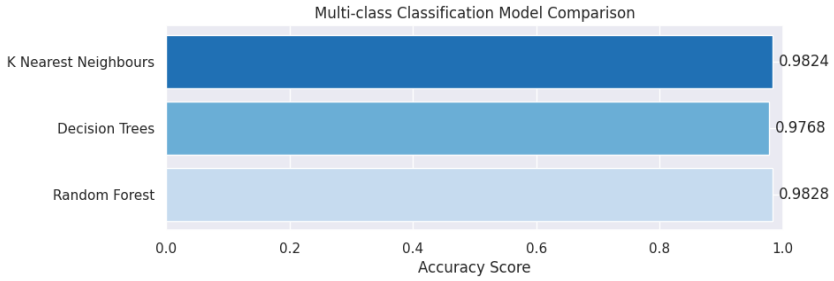
Gambar 3. 10 hasil uji performa model *random forest*

3.3.7 Langkah 7: *Comparing Multi-Class Algorithm*

Dalam rangka memperoleh model yang paling optimal dalam mendeteksi dan mengklasifikasikan berbagai jenis serangan jaringan, dilakukan perbandingan terhadap beberapa algoritma klasifikasi multikelas. Algoritma yang digunakan dalam penelitian ini meliputi:

a. *Random Forest (RF)*
b. *Decision Tree (DT)*
c. *K-Nearest Neighbors (KNN)*

Ketiga algoritma ini dipilih karena telah terbukti efektif dalam menangani masalah klasifikasi pada domain keamanan jaringan serta memiliki mekanisme pembelajaran yang berbeda, sehingga memungkinkan dilakukan analisis performa secara komprehensif. Untuk memastikan perbandingan yang adil, seluruh algoritma dilatih dan diuji pada dataset yang sama, dengan pembagian data menggunakan skema *train-test split* sebesar 80:20. Setiap model diuji menggunakan metrik evaluasi yang seragam, yaitu accuracy, *precision*, *recall*, dan *f1-score*, serta visualisasi *confusion matrix* untuk melihat akurasi per kelas.



Gambar 3. 11 Komparasi tiga algoritma Multi kelas