e-ISSN : 2722-4368

Analisis Klasifikasi Keamanan dalam Shorting Malware Android dengan Algoritma K-Nearest Neighbors

Nurcholis Majid[1], Aswan S. Sunge[2], Suherman[3]
[1,2,3]Universitas Pelita Bangsa, Jl. Inspeksi Kalimalang No.9, Cibatu, Cikarang Selatan, Bekasi, Indonesia
olismajid21@gmail.com

Abstrak. *Perkembangan teknologi dan popularitas perangkat berbasis Android telah membuka pintu lebar bagi pengembangan aplikasi yang beragam dan inovatif. Namun, kesuksesan Android juga telah menarik perhatian para penjahat cyber untuk mengembangkan Malware. Berbagai jenis perangkat dapat diinfeksi oleh malware, salah satunya adalah smartphone, dimana kasus malware terbanyak didominasi pada sistem operasi Android. Rentannya serangan malware dan dapat merugikan para pengguna Android sehingga diperlukan analisis lebih lanjut, pada kasus ini digunakan pendekatan Machine Learning untuk melakukan klasifikasi data serangan malware android. Algoritma yang digunakan adalah K-Nearest Neighbor. Mendapatkan hasil accuracy dengan nilai sebesar 98%, precision sebesar 98%, dan recall sebesar 98%. Hasil ini membuktikan bahwa algoritma K-Nearest Neighbor memberikan hasil yang cukup baik dalam mengklasifikasi malware android.*

Kata Kunci : Data Mining, Malware, Android, Klasifikasi, K-Nearest Neighbor

Abstract. *Technological developments and the popularity of Android-based devices have opened wide doors for the development of diverse and innovative applications. However, Android's success has also attracted the attention of cybercriminals to develop Malware. Various types of devices can be infected by malware, one of which is smartphones, where the majority of malware cases are dominated by the Android operating system. Malware attacks are vulnerable and can harm Android users so further analysis is needed, in this case a Machine Learning approach is used to classify Android malware attack data. The algorithm used is K-Nearest Neighbor. Get accuracy results with a value of 98%, precision of 98%, and recall of 98%. These results prove that the K-Nearest Neighbor algorithm provides quite good results in classifying Android malware.*

Keyword : Data Mining, Malware, Android, Classification, K-Nearest Neighbor

PENDAHULUAN

Perkembangan teknologi informasi telah menyentuh hampir setiap aspek kehidupan modern. Dengan mengadopsi teknologi dengan bijaksana dan berfokus pada pembangunan yang berkelanjutan dan inklusif, kita dapat mengoptimalkan manfaat teknologi informasi bagi kesejahteraan dan kemajuan masyarakat global[1]. Inovasi baru terus muncul untuk membentuk masa depan[2]. Perkembangan teknologi dan popularitas perangkat berbasis Android telah membuka pintu lebar bagi pengembangan aplikasi yang beragam dan inovatif. Namun, kesuksesan Android juga telah menarik perhatian para penjahat *cyber* untuk mengembangkan *Malware*. *Malware* Android memiliki potensi untuk mengancam privasi pengguna, menyebabkan kerugian finansial, dan bahkan merusak fungsi perangkat.

Malware telah menjadi ancaman besar bagi pengguna teknologi saat ini. Berbagai jenis perangkat dapat diinfeksi oleh *malware*, salah satunya adalah *smartphone*, di mana kasus *malware* terbanyak didominasi pada sistem operasi Android[3]. Peneliti keamanan siber kembali menemukan *malware* baru yang menyebar di Google Play Store. *Malware* baru bernama *Fleckpe* itu sudah diunduh lebih dari 620.000 kali sejak tahun 2022[4]. Data Kaspersky menunjukkan *Malware Fleckpe* sudah aktif sejak tahun lalu tapi baru kali ini ditemukan dan didokumentasikan. Kaspersky mengklaim mendeteksi dan memblokir sebanyak 79.442 serangan *malware* yang menargetkan perangkat seluler di Indonesia (tidak termasuk *adware* dan *riskware*). Kaspersky melihat kemampuan penjahat siber untuk menyebarkan elemen berbahaya ini dengan menciptakan skema yang semakin beragam[5].

Malware baru yang menyerang perangkat Android telah ditemukan oleh perusahaan keamanan Trend Micro. Memiliki nama *Guerrilla*, *malware* ini disebut telah menginfeksi jutaan perangkat Android di seluruh dunia. *Malware* ini akan mencuri informasi pribadi dari korbannya termasuk *password*, nomor kartu kredit,

hingga data sensitif lain. Selain itu, *Guerrilla* juga bisa mengakses dan mencuri data dari aplikasi apa pun yang ada di perangkat korban[6]. Rentannya serangan *malware* dan dapat merugikan para pengguna Android sehingga diperlukan analisis lebih lanjut, pada kasus ini digunakan pendekatan *Machine Learning* untuk melakukan klasifikasi data serangan *malware* android. Algoritma yang digunakan adalah *K-Nearest Neighbor*. Algoritma ini bekerja dengan cara mencari nilai K data terdekat dari data yang akan diklasifikasikan, kemudian data akan diklasifikasikan berdasarkan mayoritas kelas dari data-data terdekat tersebut[7]. Adapun beberapa referensi penelitian yang berkaitan dengan klasifikasi menggunakan algoritma *K-Nearest Neighbor* sebagai berikut:

Penelitian yang berjudul Penerapan Teknik SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Objektivitas Berita Online Menggunakan Algoritma KNN. Berdasarkan percobaan penelitian menggunakan aplikasi *weka* untuk klasifikasi dengan algoritma KNN menggunakan jumlah data sebanyak 200 data, dengan jumlah berita objektif sebanyak 176 dan berita subjektif sebanyak 24. Jumlah data yang digunakan dalam klasifikasi algoritma KNN tidak seimbang. Sehingga diperlukan tambahan teknik khusus untuk menyeimbangkan kelas data yaitu teknik *SMOTE*. Performa klasifikasi algoritma KNN yang paling baik adalah saat nilai k=9 dengan nilai akurasi, presisi, *recall* dan *F-measure* paling tinggi dari nilai k lainnya yaitu sebesar 88,00 untuk akurasi, 0,88 untuk presisi, 1 untuk *recall* dan 0,93 untuk *F-measure*. Performa klasifikasi *SMOTE*+KNN yang paling baik adalah saat nilai k=1 dengan nilai akurasi, *recall* dan *F-measure* paling tinggi dari nilai k lainnya yaitu sebesar 87,50 untuk akurasi, 0,80 untuk presisi, 0,98 untuk *recall*, dan 0,88 untuk *F-measure*[8].

Penelitian yang berjudul Algoritma K-Nearest Neighbor dengan Euclidean Distance dan Manhattan Distance untuk Klasifikasi Transportasi Bus. Dari hasil pengujian yang telah dilakuan dapat disimpulkan bahwa penerapan metode K-NN dengan metode *Euclidean Distance* dan *Manhattan Distance* diperoleh hasil akurasi tertinggi dengan nilai sebesar 84%, dengan k=3. Metode Pendekatan *Manhattan Distance* memiliki nilai selisih 2,04% lebih tinggi dibandingkan dengan *Euclidean Distance*. Persentase tersebut menunjukkan bahwa *Manhattan Distance* lebih akurat dibandingkan dengan *Euclidean Distance* sehingga *Manhattan Distance* bekerja dengan baik dalam memberikan rekomendasi untuk klasifikasi transportasi bus[9].

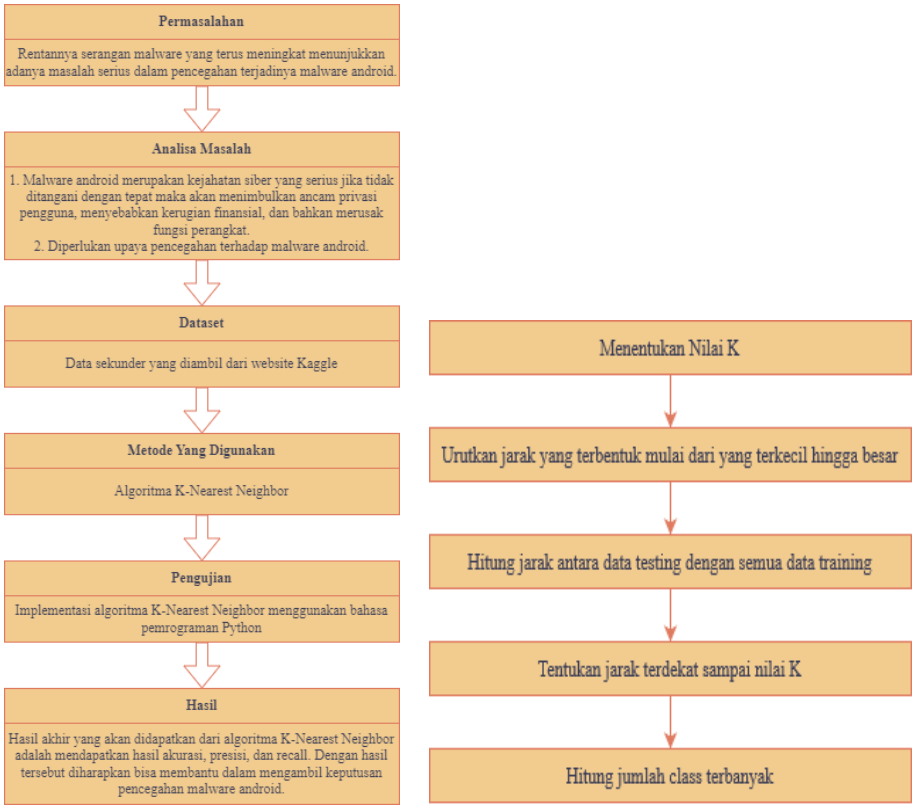
Penelitian yang berjudul Implementasi Algoritma Klasifikasi K-Nearest Neighbor (KNN) Untuk Klasifikasi Seleksi Penerima Beasiswa. Kesimpulan dari penelitian ini terciptanya aplikasi untuk seleksi data penerima beasiswa dengan menggunakan algoritma KNN. Evaluasi algoritma KNN menggunakan metode *confussion matrix* menunjukkan hasil dari perhitungan rata-rata akurasi dari metode KNN ini sebesar 90,5%. Evaluasi perbandingan data *training* dan *testing* pada metode KNN dengan *cross validation sampling* memperlihatkan hasil perhitungan rata – rata *precision* sebesar 89%[10].

Penelitian yang berjudul The Lao Text Classification Method Based on KNN. Data klasifikasi berita Laos yang digunakan dalam penelitian ini tidak seimbang, sehingga akan mempengaruhi efek klasifikasi dari *classifier*. Dalam percobaan ini, *over-sampling* digunakan untuk menyelesaikan masalah data klasifikasi yang tidak seimbang. Eksperimen menunjukkan bahwa pengambilan sampel berlebihan dapat meningkatkan hasil eksperimen. Ada tiga metode umum untuk menilai model klasifikasi: *Confusion Matrix*, *Receiver Operating Characteristic curve*, dan *Area Under Curve*. model klasifikasi masing-masing, dan kemudian menampilkan hasilnya dalam sebuah tabel. Pada test set 921 teks berita, diprediksi total 658 sampel, sehingga *Accuracy* = 658/921 = 71.4%, *Precision* = 121/161 = 75.2%, *Recall* = 121/135 = 89.6%[11].

Penelitian yang berjudul Face Recognition System Based on Kernel Discriminant Analysis, K-Nearest Neighbor and Support Vector Machine. *System of face recognition based on Kernel Discriminant Analysis* (KDA) untuk mengevaluasi kinerja sistem, eksperimen ekstensif dilakukan pada dua *database* wajah yang umum. Hasil percobaan telah membuktikan bahwa sistem yang diusulkan dapat bekerja pada kondisi yang berbeda seperti paparan pencahayaan, detail wajah, dan ekspresi wajah yang berbeda. Selain itu, hasilnya menunjukkan bahwa sistem ini memiliki tingkat pengenalan yang tinggi dan akurasi dicapai hingga 95,25% pada *database* Yale dan 96% pada *database* ORL. Di masa depan, jenis metode lain akan diterapkan untuk meningkatkan klasifikasi seperti jaringan saraf konvolusi (CNN)[12].

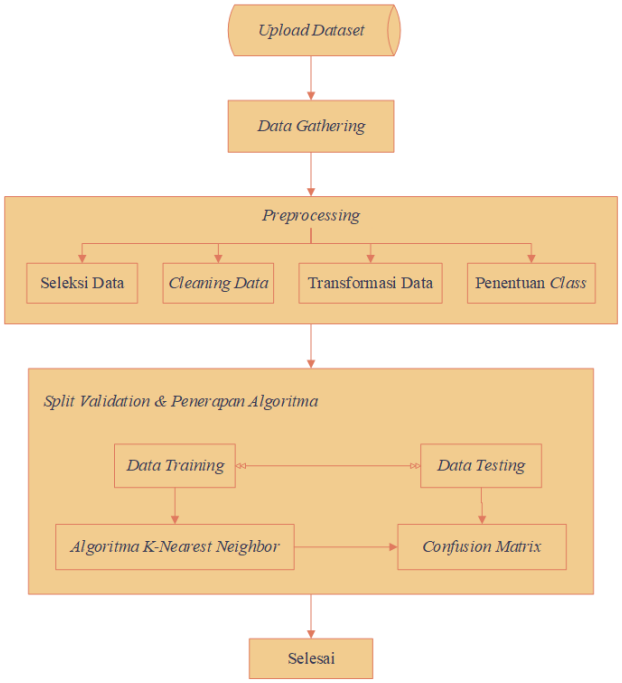
e-ISSN : 2722-4368

METODOLOGI PENELITIAN



Gambar 1. Kerangka Pemikiran dan Alur Algoritma

Berdasarkan gambar diatas didapat permasalah yang ada sebagai latar belakang penelitian. Selanjutnya analisa masalah adalah suatu proses untuk memecahkan atau membuat permasalahan lebih kompleks. Selanjutnya data yang digunakan merupakan data sekunder yang diambil melalui *Kaggle*. Selanjutnya algoritma yang akan digunakan yaitu *K-Nearest Neighbor*. Setelah itu melakukan pengujian terhadap *dataset*.



Gambar 3. Model Pengujian

2.1 Upload Dataset

Langkah pertama ialah meng-*upload dataset* yang akan digunakan.

2.2 Data Gathering

Tahap ini digunakan untuk mengumpulkan sejumlah informasi pada *dataset* mulai dari total keseluruhan data, jumlah kolom, tipe data yang digunakan, melihat jumlah variabel *NULL* atau data yang memiliki nilai kosong, dan lain-lain.

2.3 Preprocessing

Setelah *dataset* sudah di *upload* dan sudah mendapatkan informasi mengenai *dataset* yang digunakan, langkah selanjutnya adalah melakukan *preprocessing* seperti *selection data*, *cleaning data*, *transform data*, dan penentuan *class*.

2.4 *Split Validation* dan Penerapan Algoritma

Pada tahap ini data akan membagi jumlah keseluruhan *dataset* menjadi 2 yaitu data *training* dan data *testing* dengan nilai yaitu 70:30, dengan data *training* 70% dan data *testing* 30%. Setelah membagi *dataset* langkah selanjutnya adalah memasukkan data *training* kedalam model algoritma. Setelah melakukan penerapan algoritma, kemudian hasil tersebut akan diujikan dengan data *testing* sehingga akan mendapatkan bentuk *confusion matrix* sebagai nilai akhir.

HASIL DAN PEMBAHASAN

3.1 HASIL PENGUJIAN

Setelah analisis data selesai, dilanjutkan dengan pembahasan hasil dari pengujian yang dilakukan untuk mengetahui hasil nilai akurasi serta untuk mengetahui hasil klasifikasi *malware* android dengan algoritma *K-Nearest Neighbor*.

3.1.1 Import Library

Sebelum memanggil *dataset* yang akan digunakan adalah *import* dan menjalankan *library* pendukung seperti untuk menganalisis data, untuk komputasi numerik, dan untuk visualisasi data dalam bentuk grafik.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Gambar 4. *Import Library*

3.1.2 Upload Dataset

Setelah *library* yang dibutuhkan sudah dijalankan langkah selanjutnya adalah meng-*upload dataset*. Format *dataset* yang digunakan adalah CSV (*Comma Separated Value*).

```
[ ] dataset = '/content/drive/My Drive/datasetmalware.csv'
    df = pd.read_csv(dataset)
    df
```

```
<ipython-input-4-6be16789eb95>:2: DtypeWarning: Columns (92) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(dataset)
```

	transact	onServiceConnected	bindService	attachInterface	ServiceConnection	android.os.Binder	SEND_SMS	Ljava.lang.Class.getCanonicalName	Ljava.lang.Class.getMethods	Ljava.lang.Class.cast	...
0	0	0	0	0	0	0	1	0	0	0	...
1	0	0	0	0	0	0	1	0	0	0	...
2	0	0	0	0	0	0	1	0	0	0	...
3	0	0	0	0	0	0	0	0	0	1	...
4	0	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
15031	1	1	1	1	1	1	0	1	1	1	...
15032	0	0	0	0	0	0	0	0	0	1	...
15033	0	0	0	0	0	0	0	0	0	0	...
15034	1	1	1	1	1	1	0	1	1	1	...
15035	1	1	1	1	1	1	0	1	1	1	...

15036 rows × 216 columns

Gambar 5. *Dataset*

3.1.3 Data Gathering

Langkah selanjutnya adalah mengumpulkan sejumlah informasi pada *dataset* mulai dari total keseluruhan data, jumlah kolom, tipe data yang digunakan, melihat atribut unik pada data, dan lain-

lain.

1. Menampilkan informasi detail dari *dataset* mulai dari jumlah keseluruhan data, jumlah kolom, dan tipe data yang digunakan.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15036 entries, 0 to 15035
Columns: 216 entries, transact to class
dtypes: int64(214), object(2)
memory usage: 24.8+ MB
```

Gambar 6. Menampilkan Informasi Detail *Dataset*

2. Melihat isi atribut unik pada kolom *class*.

```
df['class'].unique()
```

```
array(['S', 'B'], dtype=object)
```

Gambar 7. Melihat Isi Atribut Unik Pada *Class*

3. Melihat jumlah isi atribut pada setiap *class* yang dimana B diartikan sebagai jinak dan S diartikan sebagai *malware*.

```
df.groupby(['class']).size()
```

```
class
B    9476
S    5560
dtype: int64
```

Gambar 8. Melihat Jumlah Isi Atribut Pada Setiap *Class*

4. Melihat ringkasan informasi statistik pada *dataset* dan hanya kolom yang mempunyai tipe data numerik saja yang akan ditampilkan.

```
[ ] df.describe()
```

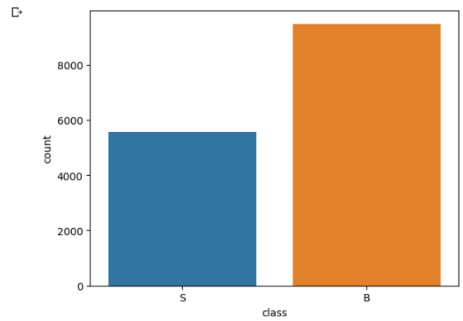
	transact	onServiceConnected	bindService	attachInterface	ServiceConnection	android.os.Binder	SEND_SMS	Ljava.lang.Class.getCanonicalName	Ljava
count	15036.000000	15036.000000	15036.000000	15036.000000	15036.000000	15036.000000	15036.000000	15036.000000	
mean	0.426443	0.446595	0.442671	0.413208	0.444932	0.486898	0.236632	0.330806	
std	0.494576	0.497156	0.496719	0.492426	0.496975	0.499845	0.425029	0.470519	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 214 columns

Gambar 9. Ringkasan Informasi Statistik *Dataset*

5. Visualisasi jumlah *class* dalam bentuk *Bar Plot*.

```
sns.countplot(x ='class', data = df)
plt.show()
```



Gambar 10. Visualisasi Jumlah *Class* Dalam Bentuk *Bar Plot*

e-ISSN : 2722-4368

3.1.4 Data Preprocessing

Pada tahap ini, data melewati proses pembersihan (*data cleaning*), transformasi data, dan seleksi data.

1. Menghapus atribut yang tidak digunakan.

```
df = df.drop('TelephonyManager.getSimCountryIso', axis=1)
```

Gambar 11. Menghapus Atribut Yang Tidak Digunakan

2. Melihat jumlah *missing values* pada setiap kolom.

```
df.isna().sum()
```

```
transact                    0
onServiceConnected          0
bindService                 0
attachInterface             0
ServiceConnection           0
                           ..
ACCESS_FINE_LOCATION        0
SET_WALLPAPER_HINTS         0
SET_PREFERRED_APPLICATIONS  0
WRITE_SECURE_SETTINGS       0
class                       0
Length: 215, dtype: int64
```

Gambar 12. Menampilkan *Missing Values*

3. Mengubah tipe data pada *class* yang sebelumnya object menjadi numerik dengan nilai 0 dan 1.

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
df['class'] = label.fit_transform(df['class'])
df.head()
```

	transact	onServiceConnected	bindService	attachInterface	ServiceConnection	android.os.Binder	SEND_SMS	Ljava.lang.Class.getCanonicalName	Ljava.lang.Class.getMethods	Ljava.l
0	0	0	0	0	0	0	1	0	0	
1	0	0	0	0	0	0	1	0	0	
2	0	0	0	0	0	0	1	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	

5 rows × 215 columns

Gambar 13. Mengubah Tipe Data Menjadi Numerik

3.1.5 Penentuan *Attribute* Dan *Class*

Menentukan nilai atribut dan *class* yang akan di tampung pada variabel X untuk atribut dan variabel Y untuk *class*.

```
X = df.iloc[:, :-1].values #digunakan sebagai atribut
Y = df.iloc[:, -1].values #digunakan sebagai class
```

Gambar 14. Penentuan *Class* dan Atribut

3.1.6 Split Validation

Langkah selanjutnya dari penelitian ini adalah membagi data menjadi dua bagian yaitu data *training* dan data *testing*. Data dibagi dengan bobot 70:30, dimana 70% digunakan sebagai data *training* dan 30% sebagai data *testing*.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
```

Gambar 15. *Split Validation*

Setelah data sudah dibagi menjadi dua lalu selanjutnya akan dicek total masing-masing data guna melihat apakah sudah sesuai atau belum.

```
print("Jumlah data training : ",len(X_train))
print("Jumlah data testing : ",len(Y_test))
```

```
Jumlah data training :  10525
Jumlah data testing :  4511
```

Gambar 16. Total Masing-Masing Data

3.1.7 Transformasi Nilai Atribut Menjadi Skala

Mentransformasi nilai atribut ke dalam bentuk skala. Fungsi utama dari *StandardScaler* adalah untuk mengubah fitur-fitur numerik dalam *dataset* menjadi distribusi normal standar dengan *mean* 0 dan simpangan baku 1.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Gambar 17. Transformasi Nilai Atribut Menjadi Skala

3.2 Pembahasan

3.2.1 Penerapan Dan Pengujian Algoritma *K-Nearest-Neighbor*

Langkah selanjutnya dalam penelitian ini adalah menerapkan algoritma *K-Nearest Neighbor* pada *dataset* yang sudah mengalami *cleaning data* dan juga pembagian data. Pada langkah ini, algoritma *K-Nearest Neighbor* diterapkan pada data *training* yang dihasilkan pada langkah sebelumnya dan menguji data *testing* dengan data *training* yang telah diterapkan pada algoritma.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, Y_train)

Y_pred = clf.predict(X_test)
```

Gambar 18. Penerapan Algoritma *K-Nearest Neighbor*

3.2.2 Hasil Pengujian Dengan Algotima *K-Nearest Neighbor*

Pada tahap ini, didapat hasil *classification report* dari hasil pengujian algoritma *K-Nearest Neighbor* untuk mengklasifikasi *dataset* yang digunakan. Hasil yang diperoleh berupa *accuracy*, *precision*, dan *recall*. Setelah itu hasil akhir juga di visualisasikan ke dalam *confusion matrix* menggunakan *library seaborn*.

Gambar 19. Hasil *Confusion Matrix*

1. *Accuracy*
   $= \frac{TP+TN}{TP+TN+FP+FN}$ x 100%
   $= \frac{2787+1635}{2787+1635+39+50}$ x 100%
   = 98.02%
2. *Precision*
   $= \frac{TP}{TP+FP}$ x 100%
   $= \frac{2787}{2787+39}$ x 100%
   = 98.61%
3. *Recall*
   $= \frac{TP}{TP+FN}$ x 100%
   $= \frac{2787}{2787+50}$ x 100%
   = 98.23%

KESIMPULAN

Berdasarkan hasil penelitian yang sudah dilakukan, dapat diambil kesimpulan bahwa penerapan algoritma *K-Nearest Neighbor* dalam mengklasifikasi *malware* android berdasarkan *Android Permission* dapat diterapkan dengan baik dan memberikan hasil yang sangat baik. Dari proses pengujian yang dilakukan mendapatkan hasil *accuracy* dengan nilai sebesar 98%, *precision* sebesar 98%, dan *recall* sebesar 98%. Dari pengujian data *testing* yang berjumlah 4511 data, diperoleh jumlah aplikasi terindeksi jinak sebesar 63% dan aplikasi terindeksi *malware* sebesar 37% sehingga dapat disimpulkan bahwa aplikasi terindeksi jinak lebih banyak dibanding aplikasi terindeksi *malware*. Dengan hasil model *machine learning* yang diterapkan diharapkan bisa membantu dalam mengambil keputusan dengan tepat untuk pencegahan *malware* android.

DAFTAR PUSTAKA

[1] Sugiarto and A. Farid, "Literasi Digital Sebagai Jalan Penguatan Pendidikan Karakter Di Era Society 5.0," *Cetta: Jurnal Ilmu Pendidikan*, vol. 6, no. 3, pp. 580–597, Jul. 2023, doi: 10.37329/cetta.v6i3.2603.
[2] Thomas Andrew, "Network Centric Warfare sebagai Upaya Transformasi Perang TNI," *DEFENDONESIA*, vol. 5, no. 1, pp. 35–45, Apr. 2021, doi: 10.54755/defendonesia.v5i1.101.
[3] A. Z. Zaidi, C. Y. Chong, Z. Jin, R. Parthiban, and A. S. Sadiq, "Touch-based continuous mobile device authentication: State-of-the-art, challenges and opportunities," *Journal of Network and Computer Applications*, vol. 191, p. 103162, Oct. 2021, doi: 10.1016/j.jnca.2021.103162.
[4] Virgina Maulita Putri, "Malware Baru Beredar di Play Store, Incar Pengguna Android di Indonesia," detik.com.
[5] Cahyandaru Kuncorojati, "Paruh Pertama 2022, Mobile Malware Masih Mengintai Indonesia," medcom.id.
[6] Agustinus Mario Damar, "Hati-Hati, Jutaan Perangkat Android Terinfeksi Malware Baru Berbahaya," liputan6.com.
[7] D. Cahyanti, A. Rahmayani, and S. A. Husniar, "Analisis performa metode Knn pada Dataset pasien pengidap Kanker Payudara," *Indonesian Journal of Data and Science*, vol. 1, no. 2, pp. 39–43, Jul. 2020, doi: 10.33096/ijodas.v1i2.13.
[8] A. N. Kasanah, M. Muladi, and U. Pujianto, "Penerapan Teknik SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Objektivitas Berita Online Menggunakan Algoritma

KNN," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 3, no. 2, pp. 196–201, Aug. 2019, doi: 10.29207/resti.v3i2.945.

[9] R. K. Dinata, H. Akbar, and N. Hasdyna, "Algoritma K-Nearest Neighbor dengan Euclidean Distance dan Manhattan Distance untuk Klasifikasi Transportasi Bus," *ILKOM Jurnal Ilmiah*, vol. 12, no. 2, pp. 104–111, Aug. 2020, doi: 10.33096/ilkom.v12i2.539.104-111.

[10] J. Homepage, S. R. Cholil, T. Handayani, R. Prathivi, and T. Ardianita, "IJCIT (Indonesian Journal on Computer and Information Technology) Implementasi Algoritma Klasifikasi K-Nearest Neighbor (KNN) Untuk Klasifikasi Seleksi Penerima Beasiswa," 2021.

[11] Z. Chen, L. J. Zhou, X. Da Li, J. N. Zhang, and W. J. Huo, "The Lao text classification method based on KNN," in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 523–528. doi: 10.1016/j.procs.2020.02.053.

[12] M. Z. Al-Dabagh, M. H. Mohammed Alhabib, and F. H. AL-Mukhtar, "Face Recognition System Based on Kernel Discriminant Analysis, K-Nearest Neighbor and Support Vector Machine," *International Journal of Research and Engineering*, vol. 5, no. 2, pp. 335–338, Mar. 2018, doi: 10.21276/ijre.2018.5.3.3.