REPOSITOR, Vol. 3, No. 3, Mei 2021, Pp. 305-314
ISSN : 2714-7975
E-ISSN : 2716-1382

Klasifikasi Malware Family Menggunakan Metode K-Nearest Neighbor (K-NN)

Achmad Rizal Yogasware*[1], Denar Regata Akbi[2], Vinna Rahmayanti Setyaning Nastiti[3]
[1,2,3]Universitas Muhammadiyah Malang
ach.rizal.yogaswara@gmail.com*[1], dnarregata@umm.ac.id[3], vinastiti@umm.ac.id[2]

Abstrak

Smartphone berbasis Android OS memiliki pengguna terbanyak saat ini karena nyaman untuk digunakan dan menawarkan berbagai fitur. Akibatnya, banyak developer malware yang menjadikan Android OS sebagai incaran utama. Setiap tahun,bermunculan jenis malware family baru yang belum dikenali. Banyak peneliti mengusulkan kerangka kerja penganalisis malware Android menggunakan teknik data mining untuk mengidentifikasi jenis malware family baru. Para peneliti memerlukan dataset Android inklusif untuk menilai penganalisis Android mereka. Pada tahun 2019, Canadian Institute for Cybersecurity (CIC) telah membuat sebuah dataset untuk umum yang diberi nama CICAndMal2019. Dataset ini dibuat dengan melakukan analisis statis dan dinamis pada smartphone yang sebenarnya. Hasil dari analisis tersebut kemudian dilakukan klasifikasi malware menggunakan matode random forest. Dalam klasifikasi malware family penelitian ini menghasilkan precision sebesar 61,2% dan recall sebesar 57,7%. Pada makalah ini, kami melakukan klasifikasi malware family dengan menggunakan dataset CICAndMal2019 menggunakan metode k-Nearest Neighbor (k-NN), hasilnya kami mendapatkan precision sebesar 83% dan recall sebesar 65%.

.

Kata Kunci: *Malware, k-Nearest Neighbor (k-NN), C5.0*

Abstract

Smartphones based on Android OS have the most users today because they are comfortable to use and offer a variety of features. As a result, many malware developers have made Android OS their main target. Every year, new types of malware families emerge that have not been recognized. Many researchers are proposing an Android malware analysis framework using data mining techniques to identify new types of malware families. The researchers needed an inclusive Android dataset to assess their Android analyzer. In 2019, the Canadian Institute for Cyber security (CIC) has created a public dataset called CICAndMal2019. This dataset is created by performing static and dynamic analysis on an actual smartphone. The results of the analysis then carried out the malware classification using the random forest method. In the classification of malware family, this study resulted in a precision of 61.2% and a recall of 57.7%. In this paper, we classify the malware family using the CICAndMal2019 dataset using the k-Nearest Neighbor (k-NN) method, the results we get a precision of 83% and a recall of 65%.

Keywords: *Malware, k-Nearest Neighbor (k-NN), C5.0*

1. Pendahuluan

Android adalah salah satu sistem operasi seluler paling populer karena fiturnya yang canggih dan menarik. Hal ini membuat para *developer malware* terdorong untuk mengembangkan atau membuat *malware* untuk sistem operasi (OS) Android [1]. *Malware* atau *Malicious Software* adalah program jahat yang berjalan di sistem komputer tanpa izin dengan tujuan merusak atau mencuri data pribadi dari sistem [2]. Perkembangan *malware* yang semakin pesat menyebabkan banyak jenis keluarga *malware* baru bermunculan. Untuk mendeteksi jenis *malware*, salah satu teknik yang dapat digunakan adalah klasifikasi *malware* menggunakan *machine learning* [3]. Klasifikasi adalah teknik pembelajaran mesin yang digunakan untuk mengidentifikasi atau memprediksi kategori data baru [4]. Ada beberapa algoritma klasifikasi dalam pembelajaran mesin seperti *k-Nearest Neighbours* (k-NN), *Support Vector Machine* (SVM), *Random Forest* dll [3].

Algoritma k-NN (k-*Nearest Neighbours*) adalah salah satu algoritma klasifikasi *machine learning* yang paling sederhana. k-NN termasuk algoritma *supervised*, di mana hasil sampel uji

baru diklasifikasikan berdasarkan sebagian besar kategori pada k-NN. Akurasi dari algoritma k-NN ditentukan oleh ada dan tidak adanya data yang tidak relevan, atau jika bobot fitur setara dengan relevansinya dengan klasifikasi [5] [6].

Penelitian tentang klasifikasi malware dengan algoritma k-NN pada machine learning sebelumnya sudah pernah dilakukan. Tahun 2019 Ginika Mahajan dkk, melakukan penelitian klasifikasi malware berdasarkan family. Dalam penelitian ini metode yang digunakan adalah dengan membandingkan 6 algoritma klasifikasi yaitu algoritma k-NN*, Decision Tree, Naive Bayes, SVM, Random Fores*t dan *Neural Network*. Hasil akhir yang didapat dari penelitian ini adalah algoritma *Neural Network* mendapat nilai akurasi terendah dengan nilai 26.03% pada tool *Knime* dan 11.6% pada *orange*. Untuk algoritma yang memiliki tingkat akurasi paling tinggi adalah *Random Forest* dengan nilai akurasi 63.49% pada *Knime* dan 94.2% pada *orange* [3].

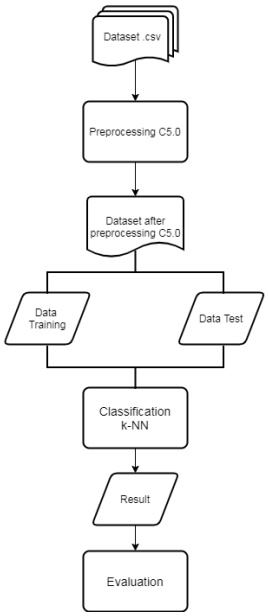
Ditahun yang sama 2019, Cho cho san dkk, melakukan penelitian klasifikasi malware pada 11 jenis keluarga malware. Untuk proses klasifikasi, algoritma yang digunakan adalah *Random Forest, k-NN (k-Nearest Neighbor)* dan *Decision Tree*. Hasil yang didapat dalam penelitian ini adalah algoritma kNN dan *Random Forest* memiliki nilai yang paling tinggi dengan nilai sama 95.8% [5].

Pada penelitian yang lain Ivan Firdausi dkk, melakukan penelitian perbandingan akurasi terhadap 5 algoritma klasifikasi *machine learning* pada 470 aplikasi yang dimana 220 *malware* dan 250 aplikasi *benign*. Algoritma klasifikasi yang dibandingkan adalah *k-Nearest Neighbors (k-NN), Naïve Bayes, J48 Decision Tree, Support Vector Machine* (SVM), dan *Multilayer Perceptron Neural Network* (MLP). Hasil yang didapat dalam penelitian ini ialah algoritma J48 *Decision Tree* menjadi algoritma yang memiliki performa terbaik dengan nilai *recall* 95.9%, *false positive rate* 2.4%, nilai *precision* 97.3%, dan nilai akurasi 96.8% [7].

Tahun 2019 Laya Taheri dkk, melakukan penelitian deteksi dan klasifikasi *malware family* menggunakan *Network Flow* dan *API-Calls*. Algoritma *machine learning* yang digunakan adalah algoritma *Random Forest*. Hasil dari penelitian ini adalah metode yang digunakan sukses meningkatkan *precision* sebesar 95,3% pada *malware Binary Classification* , *precision* sebesar 83.3% pada *Malware Category Classification* dan *precision* sebesar 59,7% pada *malware Family Classification* [8].

Pada penelitian ini, penulis ingin melakukan klasifikasi *malware* berdasarkan jenis familinya dengan data yang sama dengan yang digunakan oleh penelitan Laya Taheri dkk pada tahun 2019, dengan papernya yang berjudul "*Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls*" [8]. Algoritma klasifikasi *machine learning* yang akan digunakan adalah algoritma k-NN (*k-Nearest Neighbor*).

2. Metode Penelitian



Gambar 1. Skema penelitan

Tahapan penelitian ini dimulai dengan melakukan *preprocessing feature selectio*n pada dataset *CICInvesAndMal2019* menggunakan C5.0. Selanjutnya dataset akan dibagi menjadi 2 bagian dataset *training* dan dataset *testing.* Berikutnya klasifikasi akan dilakukan menggunakan metode k-NN dan tahap terakir adalah melakukan evaluasi pada hasil klasifikasi. Gambar 1 adalah skema pengujian yang digunakan pada penelitian ini.

2.1 Dataset *Malware Family*

Data yang digunakan dalam penelitian ini adalah *CICInvesAndMal2019* dataset yang didapat dari *website Canadian Institute for Cybersecurity* [8]. data yang didapat berupa *file* berekstensi .csv. Dataset ini terdiri dari 305 sampel *malware* yang terdiri dari 4 kategori *malware* dan 39 jenis *malware family,* seperti pada Tabel 1.

Tabel 1. Dataset CICInvesAndMal2019

No	Category	Malware Family	Number of Sample
1	Adware	Dowgin family	6
		Ewind family	10
		Feiwo family	5
		Gooligan family	10
		Kemoge family	11
		Mobidash family	8
		Selfmite family	3
		Shuanet family	7
		Youmi family	10
2	Ransomware	Charger family	10
		Jisut family	8
		Koler family	7
		LockerPin family	3
		Simplocker family	3
		Pletor family	8
		PornDroid family	7
		RansomBO family	10
		Svpeng family	6
		WannaLocker family	6
3	Scareware	AndroidDefender	16
		AndroidSpy.277 family	6
		AV for Android family	9
		AVpass family	10

		FakeApp family	4
		FakeAV family	10
		FakeJobOffer family	8
		FakeTaoBao family	7
		Penetho family	10
		VirusShield family	9
4	SMS Malware	BeanBot family	10
		Biige family	3
		FakeInst family	6
		FakeMart family	9
		Jifake family	10
		Mazarbot family	10
		Nandrobox family	10
		Plankton family	6
		SMSsniffer family	9
		Zsone family	10

2.2 *Preprocessing*

Preprocessing adalah proses pembersihan data, reduksi data, dan diskretisasi data. Fase *preprocessing* berikut akan membuat dataset lebih presisi. Kami menggunakan metode *preprocessing* data seperti penghilangan *noise* dan pengurangan atribut [9]. Metode yang digunakan dalam proses *preprocessing* adalah metode C5.0. 10-9

2.3 Algoritma C5.0

C5.0 merupakan salah satu algoritma dalam klasifikasi pada data mining dan juga termasuk dalam teknik decision tree. Algoritma ini merupakan versi pengembangan dari C4.5 yang dikembangkan oleh Ross Quinlan[10] 11-10. Atribut dengan nilai informasi terbesar akan dijadikan inti untuk node selanjutnya. Informasi sangat diperlukan untuk mengklasifikasi tiap sampel yang digunakan dengan cara Persamaan 1. Setelah itu, nilai himpunan bagian dari atribut A akan dihitung menggunakan Persamaan 2. Kemudian untuk perolehan informasi dihitung menggunakan Persamaan 3 [11] 12-11.

$$I\,(s_1,\ s_2,\ \ldots,\ s_m) = -\sum_{i=1}^{m} p_i \log_2\ (p_i) \tag{1}$$

$$E(A) = \sum_{j=1}^{y} \frac{s_{1j}+\cdots+s_{mj}}{s}\ I\ (s_{1j}, \ldots, s_{mj}) \tag{2}$$

$$Gain(A) = I\ (s_1,\ s_2,\ \ldots,\ s_m) - E(A) \tag{3}$$

Keterangan :
S : jumlah data sampel
Si : jumlah sampel dalam S di Ci class
m : jumlah atribut class
pi : pembagian class dan estimasi menggunakan s i/s
E(A) : nilai bagian dari himpunan atribut A
Sj : sampel dari S yang didapatkan dari nilai A
Sij : jumlah sampel di class Ci dari himpunan Sj
Gain (A) : informasi yang didapatkan dari atribut A

2.4 Pembagian Dataset

Dataset akan dibagi menjadi dataset *training* dan dataset *testing.*Dalam penelitian ini akan ada 4 tipe pembagian dataset seperti pada Tabel 2.

Tabel 2. Persentase pembagian dataset

Percentage	Training Data	Test Data
60% : 40%	283 malware	122 malware
70% : 30%	213 malware	92 malware
80% : 20 %	244 malware	61 malware
90% : 10%	274 malware	31 malware

2.5 k-Nearest Neighbor (k-NN)

Algoritma *k-Nearest Neighbor* adalah algoritma *supervised learning* yang menggunkan konsep ketetanggaan dimana hasil dari data yang baru diklasifikasikan berdasarkan mayoritas tetangga terdekat (k) [12]9-12. Algoritma kNN berkerja dengan cara menemukan k tetangga terdekat dari data yang belum dikeahui kelasnya. Penentuan k tetangga terdekat dilakukan dengan menghitung jarak dari data uji ke setiap data latih yang ada. Selanjutnya dipilih sejumlah k data yang memiliki jarak terdekat. Kelas dari data uji ditentukan dari mayoritas kelas k data latih terdekat. Perhitungan jarak pada k-NN terdapat beberapa metode seperti *euclidean distance, Manhattan distance* dan *Mahalanobis distance*[13]11-13. Penelitian ini menggunakan metode *euclidean distance* untuk perhitungan jarak tetangga, seperti pada Persamaan 4.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(xi - yi)}^{2} \quad (4)$$

3. Hasil Penelitian dan Pembahasan

Dataset yang sudah melewati proses *preprocessing feature selection* memiliki atribut yang lebih sedikit, dimana dataset sebelum *preprocessing* mempunyai 918 atribut dan setelah *preprocessing* mempunyai 129 atribut. 129 atribut yang berpengaruh besar dalam klasifikasi dapat dilihat pada Tabel 3.

Tabel 3. Atribut dataset setelah preprocessing

No	Percentage	*attribute*
1.	100.00%	X2Grammed_APICalls_.7
2.	100.00%	X2Grammed_APICalls_.53
3.	100.00%	X2Grammed_APICalls_.416
4.	100.00%	Min_Packet_Length
5.	100.00%	PSH_Flag_Count
6.	100.00%	Down.Up_Ratio
7.	97.30%	Bwd_Packets.s
8.	94.05%	X2Grammed_APICalls_.142
9.	94.05%	X2Grammed_APICalls_.411
10.	94.05%	Fwd_Packets.s
11.	82.70%	X2Grammed_APICalls_.357
12.	81.08%	X2Grammed_APICalls_.41
13.	81.08%	X2Grammed_APICalls_.50
14.	81.08%	X2Grammed_APICalls_.329

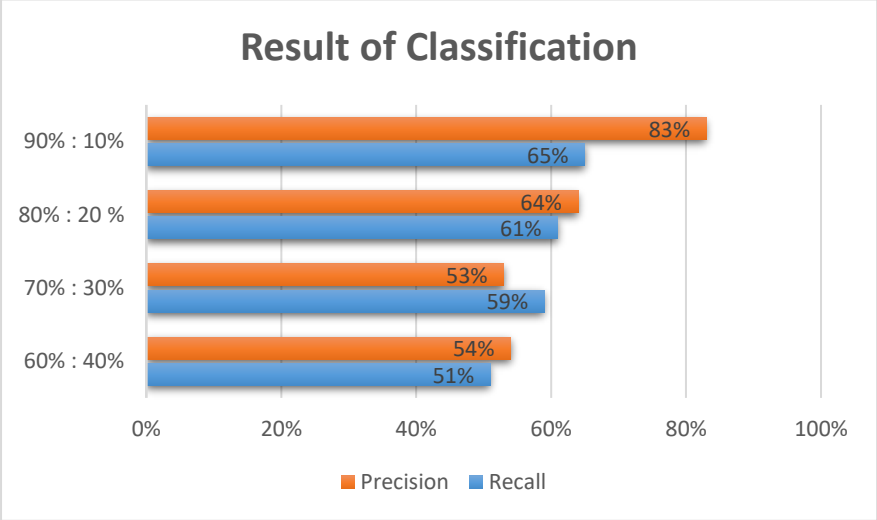
15.	80.00%	X2Grammed_APICalls_.30
16.	75.68%	X2Grammed_APICalls_.301
17.	71.89%	X2Grammed_APICalls_.59
18.	66.49%	X2Grammed_APICalls_.263
19.	65.41%	X2Grammed_APICalls_.125
20.	64.32%	X2Grammed_APICalls_.342
21.	63.78%	X2Grammed_APICalls_.364
22.	61.62%	Fwd_PSH_Flags
23.	57.84%	Total_Length_of_Fwd_Packets
24.	57.84%	Idle_Min
25.	55.68%	X2Grammed_APICalls_.109
26.	55.14%	X2Grammed_APICalls_.217
27.	53.51%	X2Grammed_APICalls_.796
28.	52.97%	Fwd_IAT_Min
29.	52.43%	X2Grammed_APICalls_.90
30.	49.19%	X2Grammed_APICalls_.1
31.	47.57%	X2Grammed_APICalls_.255
32.	44.32%	X2Grammed_APICalls_.532
33.	40.00%	act_data_pkt_fwd
34.	38.92%	X2Grammed_APICalls_.48
35.	35.68%	X2Grammed_APICalls_.160
36.	35.68%	Idle_Std
37.	35.14%	X2Grammed_APICalls_.408
38.	34.05%	X2Grammed_APICalls_.4
39.	34.05%	X2Grammed_APICalls_.89
40.	33.51%	X2Grammed_APICalls_.331
41.	31.89%	X2Grammed_APICalls_.14
42.	31.89%	X2Grammed_APICalls_.38
43.	31.35%	X2Grammed_APICalls_.116
44.	31.35%	Active_Std
45.	29.19%	X2Grammed_APICalls_.138
46.	28.65%	X2Grammed_APICalls_.22
47.	28.65%	X2Grammed_APICalls_.34
48.	28.11%	X2Grammed_APICalls_.111
49.	27.57%	X2Grammed_APICalls_.5
50.	27.57%	X2Grammed_APICalls_.35
51.	27.03%	X2Grammed_APICalls_.86
52.	25.95%	X2Grammed_APICalls_.193
53.	25.95%	X2Grammed_APICalls_.222
54.	25.95%	X2Grammed_APICalls_.353
55.	25.41%	X2Grammed_APICalls_.121
56.	25.41%	X2Grammed_APICalls_.296
57.	25.41%	X2Grammed_APICalls_.344
58.	24.86%	X2Grammed_APICalls_.240
59.	24.86%	X2Grammed_APICalls_.283
60.	24.86%	X2Grammed_APICalls_.303
61.	23.78%	X2Grammed_APICalls_.13
62.	23.78%	X2Grammed_APICalls_.363
63.	23.24%	X2Grammed_APICalls_.99
64.	23.24%	X2Grammed_APICalls_.320
65.	22.70%	X2Grammed_APICalls_.2
66.	22.16%	X2Grammed_APICalls_.12
67.	22.16%	Flow_Packets.s
68.	21.62%	X2Grammed_APICalls_.24
69.	21.62%	X2Grammed_APICalls_.66
70.	21.62%	X2Grammed_APICalls_.309
71.	21.62%	Flow_Bytes.s
72.	21.62%	Init_Win_bytes_backward

73.	21.08%	X2Grammed_APICalls_.213
74.	21.08%	Fwd_Packet_Length_Min
75.	21.08%	Idle_Max
76.	20.54%	X2Grammed_APICalls_.56
77.	20.00%	X2Grammed_APICalls_.264
78.	19.46%	X2Grammed_APICalls_.101
79.	19.46%	X2Grammed_APICalls_.108
80.	19.46%	X2Grammed_APICalls_.343
81.	18.92%	X2Grammed_APICalls_.265
82.	18.38%	X2Grammed_APICalls_.77
83.	17.30%	X2Grammed_APICalls_.0
84.	17.30%	X2Grammed_APICalls_.74
85.	17.30%	X2Grammed_APICalls_.170
86.	16.22%	X2Grammed_APICalls_.225
87.	15.68%	X2Grammed_APICalls_.3
88.	15.68%	X2Grammed_APICalls_.211
89.	15.14%	X2Grammed_APICalls_.28
90.	14.59%	X2Grammed_APICalls_.17
91.	14.59%	X2Grammed_APICalls_.40
92.	14.59%	X2Grammed_APICalls_.192
93.	13.51%	X2Grammed_APICalls_.8
94.	12.97%	X2Grammed_APICalls_.18
95.	12.97%	X2Grammed_APICalls_.165
96.	12.97%	X2Grammed_APICalls_.197
97.	12.97%	X2Grammed_APICalls_.293
98.	12.97%	URG_Flag_Count
99.	12.43%	Bwd_IAT_Mean
100.	11.89%	X2Grammed_APICalls_.311
101.	10.27%	X2Grammed_APICalls_.221
102.	10.27%	Flow_Duration
103.	9.73%	ACK_Flag_Count
104.	9.19%	X2Grammed_APICalls_.36
105.	9.19%	X2Grammed_APICalls_.393
106.	9.19%	X2Grammed_APICalls_.524
107.	8.65%	X2Grammed_APICalls_.19
108.	8.65%	X2Grammed_APICalls_.64
109.	8.65%	X2Grammed_APICalls_.84
110.	8.11%	X2Grammed_APICalls_.281
111.	7.57%	X2Grammed_APICalls_.137
112.	7.57%	X2Grammed_APICalls_.209
113.	7.57%	X2Grammed_APICalls_.220
114.	7.03%	X2Grammed_APICalls_.43
115.	7.03%	X2Grammed_APICalls_.91
116.	7.03%	X2Grammed_APICalls_.230
117.	7.03%	X2Grammed_APICalls_.328
118.	7.03%	Bwd_IAT_Total
119.	6.49%	X2Grammed_APICalls_.171
120.	5.95%	X2Grammed_APICalls_.51
121.	5.95%	X2Grammed_APICalls_.87
122.	5.95%	X2Grammed_APICalls_.205
123.	5.95%	X2Grammed_APICalls_.368
124.	5.41%	X2Grammed_APICalls_.212
125.	4.32%	X2Grammed_APICalls_.39
126.	4.32%	X2Grammed_APICalls_.229
127.	4.32%	Fwd_IAT_Total
128.	3.78%	X2Grammed_APICalls_.275
129.	3.78%	X2Grammed_APICalls_.663

Data yang sudah didapat dari proses *preprocessing feature selection* selanjutnya dilakukan klasifikasi menggunakan metode k-NN. Nilai k yang digunakan untuk k-NN adalah 3. Hasil klasifikasi dapat dilihat pada Tabel 4 dan Gambar 2.

Tabel 4. Hasil klasifikasi k-NN

Percentage	Recall	Precision
60% : 40%	51%	54%
70% : 30%	59%	53%
80% : 20 %	61%	64%
90% : 10%	65%	83%



Gambar 2. Grafik hasil klasifikasi k-NN

Tabel 4 dan Gambar 2 menunjukkan bahwa persentase pembagian dataset dari 90% banding 10% memiliki hasil terbaik dengan nilai *recall* sebesar 65% dan *precision* sebesar 83%. Hal ini dikarenakan *machine learning* memiliki banyak data latih sehingga keandalan *machine learning* dalam proses klasifikasi dari *malware family* semakin meningkat.

4. Kesimpulan

Teknik klasifikasi untuk *malware family* menggunakan k-NN dapat memberikan hasil yang baik dengan pembagian data *training* dan data *testing* yang tepat. Komposisi pembagian data 90% data *training* dan 10% data *testing* memiliki hasil terbaik dengan nilai *recall* 65% dan *precision* 83%. Hal ini juga menunjukkan bahwa *preprocessing* dataset menggunakan algoritma C5.0 dapat meningkatkan hasil klasifikasi.

Referensi

[1] Rajkumar, M. N., Kumar, V. V., & Vijayabhasker, R. (2020). *A Hybrid Approach to Detect the Malicious Applications in Android-Based Smartphones Using Deep Learning.* In Handbook of Research on Machine and Deep Learning Applications for Cyber Security (pp. 176-194). IGI Global.

[2] Udayakumar, N., Subbulaksmi, T., Mishra, A., Mishra, S., & Jain, P. (2019). *Malware Category Prediction Using KNN and SVM Classifiers*. International Journal of Mechanical Engineering and Technology (IJMET) (pp.787-797).

[3] Mahajan, G., Saini, B., & Anand, S. (2019, February). *Malware Classification Using Machine Learning Algorithms and Tools*. In 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP) (pp. 1-8). IEEE.

[4] Milosevic, N., Dehghantanha, A., & Choo, K. K. R. (2017). *Machine learning aided Android malware classification.* Computers & Electrical Engineering, 61, 266-274.

[5] San, C. C., Thwin, M. M. S., & Htun, N. L. (2019). *Malicious software family classification using machine learning multi-class classifiers*. In Computational Science and Technology (pp. 423- 433). Springer, Singapore.

[6] Kruczkowski, M., & Niewiadomska-Szynkiewicz, E. (2014). *Comparative study of supervised learning methods for malware analysis.* Journal of Telecommunications and Information Technology.

[7] Firdausi, I., Erwin, A., & Nugroho, A. S. (2010, December). *Analysis of machine learning techniques used in behavior-based malware detection.* In 2010 Second international conference on advances in computing, control, and telecommunication technologies (pp. 201-203). IEEE.

[8] Taheri, L., Kadir, A. F. A., & Lashkari, A. H. (2019, October). *Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls*. In 2019 International Carnahan Conference on Security Technology (ICCST) (pp. 1-8). IEEE.

[9] Chandrasekar, P., & Qian, K. (2016, June). *The impact of data preprocessing on the performance of a naive bayes classifier*. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 2, pp. 618-619). IEEE.

[10] Kurniawan, E., Nhita, F., Aditsania, A., & Saepudin, D. (2019, July). *C5. 0 algorithm and synthetic minority oversampling technique (SMOTE) for rainfall forecasting in Bandung regency.* In 2019 7th *International Conference on Information and Communication Technology* (ICoICT) (pp. 1-5). IEEE

[11] Sandag, G. A., Leopold, J., & Ong, V. F. (2018). K*lasifikasi Malicious Websites Menggunakan Algoritma K-NN Berdasarkan Application Layers dan Network Characteristics*. CogITo Smart Journal, 4(1), 37-45.

[12] Hackeling, G. (2017). *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd.