Membangun Sistem Deteksi Keylogger dan Klasifikasi berbasis machine learning

Ade Lailani (20920003)
SK6091-Independent Research in Computational Science
Sains Komputasi–Institut Teknologi Bandung

Abstrak.
Keyloggers adalah salah satu serangan *cyber* yang aktif berkembang didunia maya dengan tujuan untuk mencuri uang, data keuangan, kekayaan intelektual, atau sekadar mengganggu operasi perusahaan tertentu. *Machine learning* terus dikembangkan dalam penelitian di bidang analisis *keylogger*. Telah dilakukan membangun sistem deteksi *keylogger* dan klasifikasi dengan menyajikan metode dengan parameter tertentu yang direkomendasikan berbasis machine learning. Hasil akhir didapat parameter yang menghasilkan nilai terbaik yang ditunjukan dengan nilai accuracy, precision, recall, dan F1-score yang tinggi. Pada sistem deteksi keylogger dengan metode ANN didapat parameter terbaik yaitu fungsi aktivasi output sigmoid, optimizer adam dan hidden layer 64-64-64 dengan nilai akurasi 71%. Pada sistem klasifikasi keylogger didapat parameter terbaik yaitu pada metode KNN parameter manhattan distance (p=1) dan n =3 dengan nilai akurasi 89%. Sementara pada metode Decision Tree dengan parameter time size 0.05, didapatkan nilai akurasi 99%.

Keyword : keylogger, ANN, KNN, Decision tree.

1. PENDAHULUAN

Serangan *Cyber* pada dunia maya adalah operasi Internet berbahaya yang diluncurkan sebagian besar oleh organisasi kriminal yang tujuannya mungkin untuk mencuri uang, data keuangan, kekayaan intelektual, atau sekadar mengganggu operasi perusahaan tertentu.

Keyloggers adalah salah satu ancaman yang aktif berkembang terhadap kerahasiaan pengguna karena dapat dijalankan di ruang pengguna, dengan mudah didistribusikan dan mengunggah informasi ke server jarak jauh. *Keylogger* adalah jenis *spyware* yang digunakan untuk merekam penekanan tombol yang dilakukan oleh pengguna. dalam kebanyakan kasus, penjahat dunia maya menggunakan keylogger untuk mengamati penekanan tombol korban dan mengambil informasi dari komputer mereka atau perangkat komputasi lainnya.

Keylogger umumnya terdiri dari dua kategori [1]. kategori pertama yaitu sistem *keyloggers* disuntikkan ke perangkat pengguna berupa flasdisk untuk mencuri informasi melalui penekanan tombol. kategori kedua adalah pencurian kredensial seperti ID pengguna dan kata sandi yang mereka gunakan untuk mengetik untuk masuk ke situs web atau aplikasi apa pun di dunia maya.

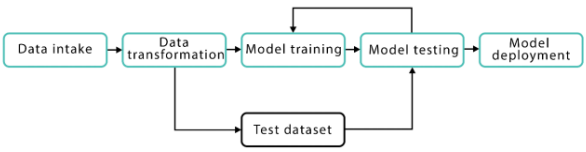
Artificial Neural Network (ANN) adalah salah satu algoritma pembelajaran mesin yang paling populer, dengan aplikasi area luas dalam pemodelan prediktif maka metode ANN akandigunakan pada penelitian ini untuk sistem deteksi *keylogger*. Sementara untuk klasifikasi metode yang sering digunakan diantaranya *K-nearest neighbours (KNN)* dan *Decision Tree*.

Tujuan atau luaran dari penelitian ini yaitu menyajikan metode dengan parameter tertentu yang direkomendasikan untuk deteksi dan klasifikasi *keylogger* berbasis *machine learning*. Kedepannya diharapkan penelitian yang dilakukan dapat bermanfaat sebagai dasar untuk penelitian lebih lanjut di bidang analisis *keylogger* dengan metode *machine learning*.

2. TEORI DASAR

2.1 MECHINE LEARNING

Ide dasar dari setiap tugas pembelajaran mesin adalah untuk melatih model, berdasarkan beberapa algoritma, untuk melakukan tugas tertentu: klasifikasi, klasterisasi, regresi, dll. Pelatihan dilakukan berdasarkan dataset input, dan model yang dibangun selanjutnya digunakan untuk membuat prediksi.



Gambar 1. Alur kerja umum machine learning [2]

Dari perspektif mechine learning, deteksi keylogger dapat dilihat sebagaimasalah klasifikasi atau klasterisasi. jenis keylogger yang tidak diketahui harus dikelompokkan ke dalam beberapa klaster, berdasarkan properti tertentu, yang diidentifikasi oleh algoritme.

2.2 *ARTIFICIAL NEURAL NETWORK (ANN)*

Artificial Neural Network (ANN) adalah salah satu algoritma pembelajaran mesin yang paling populer, dengan aplikasi area luas dalam pemodelan prediktif dan pengklasifikasi. Saat ini,

banyak model lanjutan dari *Neural Network* seperti *Convolutional Neural Network*, model pembelajaran dalam yang populer di domain visi komputer, keamanan jaringan, kecerdasan buatan, aplikasi robotika, perawatan kesehatan dan banyak lagi teknologi canggih [3]. ANN dapat diaplikasikan untuk prediksi dengan menggunakan *Keras*. *Keras* adalah API pembelajaran mendalam yang ditulis dengan Python, berjalan di atas platform pembelajaran mesin *TensorFlow*. Ini dikembangkan dengan fokus pada memungkinkan eksperimen cepat [4].

ANN dianggap sebagai model yang membutuhkan proses pelatihan agar dapat melakukan prediksi kelas suatu data uji baru yang ditemukan.

ANN ditentukan oleh 3 hal yaitu [5] :

a. Pola hubungan antar neuron (arsitektur jaringan)
b. Metode untuk menentukan bobot penghubung (metode training/learning/ algoritma).
c. Fungsi Aktivasi

2.3 K-NEAREST NEIGHBORS (KNN)

Algoritme *K-Nearest Neighbors* merupakan pendekatan untuk mencari kasus dengan menghitung kedekatan antara kasus yang baru dengan kasus lama berdasarkan pencocokan bobot dari sejumlah fitur yang ada. Kedekatan biasa berada pada nilai 0 s.d 1, nilai 0 artinya kedua kasus mutlak tidak mirip, sedang nilai 1 kasus mirip mutlak [6] Proses perhitungan kedekatan antara dua kasus dilakukan dengan persamaan (1) berikut ini.

$$similarity(T,S) = \frac{\sum_{i=1}^{n} f(T_i,S_i)*W_i}{W_i} \quad (1)$$

Keterangan:
T : kasus baru
S : kasus yang ada dalam penyimpanan n : jumlah atribut dalam setiap kasus
I : atribut individu antara 1 s.d. n
f : fungsi similarity atribut i antara kasus T dan kasus S
w : bobot yang diberikan pada atribut ke-i

2.4 DECISION TREE

Decision Tree atau pohon keputusan merupakan metode klasifikasi dan prediksi yang sangat kuat dan terkenal. Fakta yang ada akan diubah menjadi pohon keputusan yang merepresentasikan aturan. Pohon keputusan juga dapat digunakan untuk meheksplorasi data, menemukan hubungan tersembunyi antara sejumlah variabel input dengan target [6].

Terdapat beberapa algoritme dalam pembentukan pohon keputusan antara lain ID3, CART dan C4.5. C4.5 merupakan pengembangan dari algoritme ID3. Secara umum algoritme C4.5 untuk membangun pohon keputusan adalah sebagai berikut:

1) Pilih atribut sebagai akar
2) Buat cabang untuk tiap nilai
3) Bagi kasus dalam cabang
4) Ulangi proses setiap cabang sampai semua kasus memiliki kelas yang sama

Untuk memilih atribut akar didasarkan pada gain ratio tertinggi dari atribut ang ada. Proses perhitungan gain dilakukan dengan persamaan (2) berikut ini.

$$Gain(S,A) = Entropy(S) - \sum_{i=1}^{n} \frac{S_i}{S} * Entropy\ (S_i) \quad (2)$$

Keterangan:

S : himpunan kasus
A : atribut
n : jumlah partisi atribut A
|Si| : jumlah kasus pada partisi ke-i
|S| : jumlah kasus dalam S

Sedangkan proses perhitungan nilai entropi dilakukan dengan persamaan (2) berikut ini.

$$Entropy\ (S) = \sum_{i=1}^{n} - pi * log_2\ pi \quad (3)$$

Keterangan:

S : himpunan kasus
A : fitur
N : jumlah partisi S
pi : proporsi dari Si terhadap S

Pengukuran kinerja klasifikasi dilakukan dengan mengevaluasi hasil pengujian menggunakan matriks konfusi atau confusion matrix. Confusion Matrix merupakan metode untuk mengevaluasi model klasifikasi untuk memperkirakan objek yang benar atau salah [7]. Pada Tabel 1 berikut diberikan matriks konfusi dua kelas:

Tabel 1. Tabel Confusion Matrix Dua Kelas

Clasification	*Predicted Class*	
	Class=Yes	*Class=No*
Class=Yes	a (True Positive)	b (False Negative)
Class=No	c (False Positive)	d (True Negative)

Pada tabel di atas, true positive (TP) adalah jumlah record positif yang diklasifikasikan sebagai positif, false positive (FP) adalah jumlah record negatif yang diklasifikasikan sebagai positif, false negatives (FN) adalah jumlah record positif yang diklasifikasikan sebagai negatif, true negatives

(TN) adalah jumlah record negatif yang diklasifikasikan sebagai negatif. Semakin tinggi nilai TP dan TN semakin baik pula tingkat klasifikasi dari akurasi, presisi, *recall* dan F1-score. Akurasi merupakan tingkat kedekatan antara nilai prediksi dengan nilai sebenarnya. Presisi menunjukan tingkat ketepatan atau ketelitian dalam pengklasifikasian. Sedangkan *recall* berfungsi untuk mengukur proporsi positif aktual yang benar diidentifikasi. *F1-Score* merupakan perbandingan rata-rata presisi dan recall yang dibobotkan

Rumus yang digunakan dalam pengukuran performance ditunjukkan sebagai berikut:

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

$$\text{Pr}ecission = \frac{TP}{TP + FP} \quad (5)$$

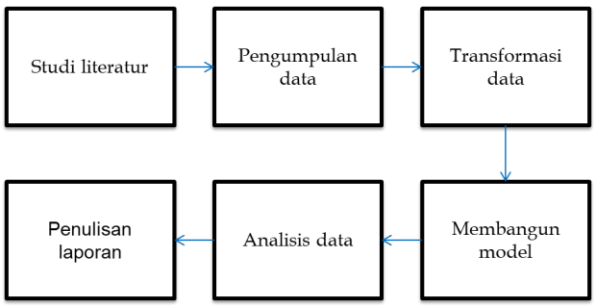
$$\text{Re}call = \frac{TP}{TP + FN} \quad (6)$$

$$F1 - score = 2 \times \frac{(\text{Re}call \times \text{Pr}ecission)}{(\text{Re}call + \text{Pr}ecission)} \quad (7)$$

3. METODOLOGI

3.1 ALUR PENELITIAN

Dalam studi ini, Komputasi dilakukan dengan menggunakan Python 3.7. pemodelan di lakukan dengan mengimplementasikan MLP *Backpropagation* pada Python menggunakan *Keras* dengan *backend Tensorflow* untuk deteksi dan klasifikasi *keylogger*. Berikut adalah alur penelitian yang akan dilakukan.



Gambar 2. Alur penelitian

a) Pengumpulan data

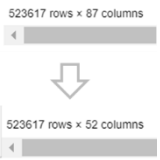
Data saat ini (berasal dari situs web CIC) [8] berisi 523617 sampel *keylogger* dan sampel benign. Pembagian kelasnya adalah Data benign dengan 309415 Observations dan Data Keylogger dengan 214202 Observations Dengan 86 columns.

b) Transformasi Data

Pada tahap ini, data sudah didapatkan ditransformasi, dibersihkan, dan dinormalisasi agar cocok untuk algoritma. Data dikonversi sehingga terletak pada range yang sama, memiliki format yang sama, dll. Berikut adalah beberapa langkah yang dilakukan pada transformasi data.

Gambar 3. mengubah data mentah menjadi angka

Gambar 4. mengubah nama kelas dari benign dan keylogger manjadi anka 0 dan 1



Gambar 5. membersihkan data

membersihkan data dengan menghapus beberapa kolom yang memiliki banyak kekosongan. Data baru disimpan untuk digunakan untuk sistem deteksi dan klasifikasi. Data yang baru berisi jumlah data benign dan keylogger sebagai berikut:

```
0    308813
1    214804
Name: Class_num, dtype: int64
```

Gambar 6. Jumlah data benign dan keylogger baru

Pada akhir tahap transformasi data dilakukan split data. Memisahkan dataset menjadi train, test dan validasi set dengan rasio 70%:5%:15%.

3.2 MEMBANGUN MODEL

a) ANN

Berikut adalah model *Artificial Neural Network* yang dibangun untuk keperluan deteksi keylogger. Parameter penting yang digunakan dalam model ini diantaranya yaitu *fungsi aktivasi output*, *optimzer*, dan *hidden layer*.

```
clf = models.Sequential()
clf.add(layers.Dense(64, activation='relu', input_dim=data_scaled.shape[1]-1))
clf.add(layers.Dense(64, activation='relu'))
clf.add(layers.Dense(64, activation='relu'))
clf.add(layers.Dense(data_scaled.Class_num.nunique(), activation='softmax'))

clf.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 64)                3328
_________________________________________________________________
dense_5 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_6 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 130
=================================================================
Total params: 11,778
Trainable params: 11,778
Non-trainable params: 0
```

```
clf.compile(optimizer='adam',
 loss='categorical_crossentropy',
 metrics=[metrics.CategoricalAccuracy()])
```

Gambar 7. Model ANN

b) KNN

Berikut adalah model KNN yang dibangun untuk keperluan klasifikasi *keylogger*. Parameter penting yang digunakan dalam model ini diantaranya yaitu *n_neighborn* dan parameter *p*.

```
: from sklearn.neighbors import KNeighborsClassifier
  clf = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=1)
  clf.fit(X_train, y_train)
```

```
: KNeighborsClassifier(p=1)
```

Gambar 8. Model KNN

c) Decision tree

Berikut adalah model *decision tree* yang dibangun untuk keperluan klasifikasi *keylogger*. Parameter penting yang digunakan dalam model ini yaitu *test_size*.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# we're scaling here in order to visualize it easily
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion='entropy', random_state=0)
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Gambar 9. Model decision tree

3.3 PARAMETER

a) ANN

Parameter *fungsi aktivasi output*, *optimzer*, dan *hidden layer* akan divariasikan untuk mendapatkan hasil akhir dengan parameter yang menghasilkan nilai terbaik. Berikut adalah variasi parameter *fungsi aktivasi output*, *optimzer*, dan *hidden layer* yang akan diuji.

Tabel 2. variasi parameter fungsi aktivasi output, optimzer, dan hidden layer pada model ANN

Parameter	Variasi
Fungsi aktivasi	• Sigmoid • Softmax
Optimizer	• Adam • Adadelta • RMSprop • SGD
Hiden layer	• 64-64 • 40-40-20 • 64-64-64 • 65-5-25 • 64-64-64-64 • 65-5-65-5

b) KNN

Parameter yang akan diujikan dalam model KNN diantaranya yaitu *n_neighborn* dan parameter *p*. Berikut adalah variasi parameter yaitu *n_neighborn* dan parameter *p* yang akan diuji.

Tabel 3. variasi parameter yaitu n_neighborn dan parameter p yang akan diuji pada model KNN

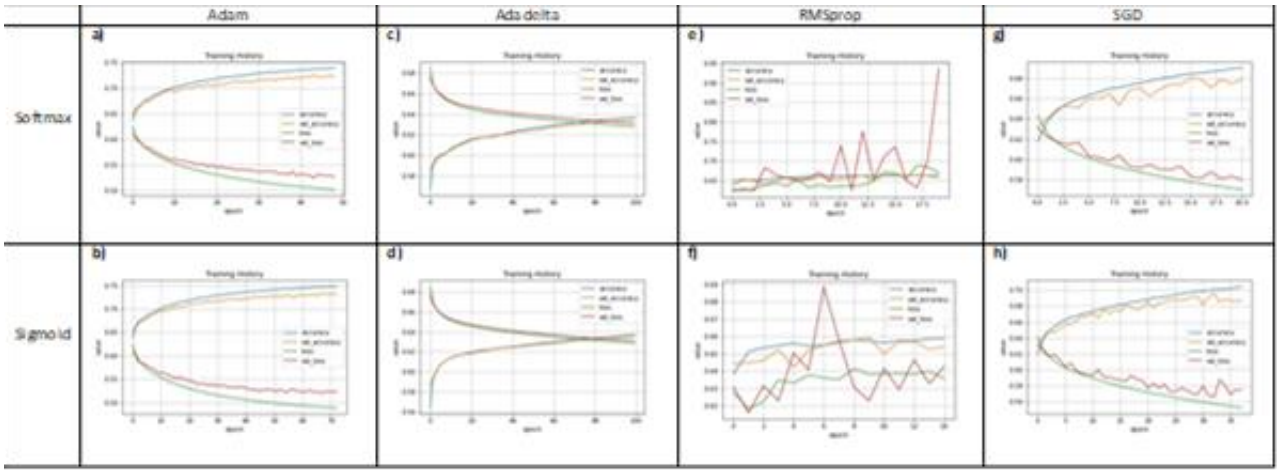
Parameter	variasi
p (distance)	• p = 1 (manhattan) • p = 2 (Euclidean)
n_neighborn	• n = 3 • n= 5 • n=7

c) Decision tree

Parameter yang akan diujikan dalam model *decision tree* yaitu variasi *test_size* sebagai berikut.

Tabel 4. variasi parameter test_size yang akan diuji pada model decision tree.

Parameter	variasi
test_size	• 0.05 • 0.25 • 0.5



Gambar 10.grafik variasi fungsi aktivasi dan opimizer model ANN;a)softmax-adam ; b)sigmoid-adam; c)softmax-adadelta; d)sigmoid-adadela; e)softmax-RMSprop; f)sigmoid-RMSprop; g)softmax-SGD; h)sigmoid-SGD

4. HASIL DAN ANALISIS

4.1 SISTEM DETEKSI DENGAN ANN

Berikut adalah hasil uji model dengan Parameter *fungsi aktivasi output* dan *optimzer* yang divariasikan untuk mendapatkan hasil akhir dengan parameter yang menghasilkan nilai terbaik.

Tabel 5. Hasil dengan variasi parameter fungsi aktivasi output dan optimzer pada model ANN

	Sigmoid		Softmax	
Adam	loss	0.533425	loss	0.519783
	categorical_accuracy	0.716361	categorical_accuracy	0.725662
	val_loss	0.548968	val_loss	0.540701
	val_categorical_accuracy	0.706139	val_categorical_accuracy	0.714654
Adadelta	loss	0.638904	loss	0.639640
	categorical_accuracy	0.622904	categorical_accuracy	0.625336
	val_loss	0.641511	val_loss	0.641418
	val_categorical_accuracy	0.622018	val_categorical_accuracy	0.624632
RMSprop	loss	0.647489	loss	0.634530
	categorical_accuracy	0.658704	categorical_accuracy	0.655052
	val_loss	0.683654	val_loss	0.639201
	val_categorical_accuracy	0.655612	val_categorical_accuracy	0.652265
SGD	loss	0.592345	loss	0.579169
	categorical_accuracy	0.670779	categorical_accuracy	0.681931
	val_loss	0.598273	val_loss	0.588644
	val_categorical_accuracy	0.664207	val_categorical_accuracy	0.674887

Pada tabel dan gambar didapatkan hasil paremeter optimizer dengan nilai akurasi tertinggi yaitu optimizer Adam di ikuti SGD, RMSprop, dan terakhir Adadelta.

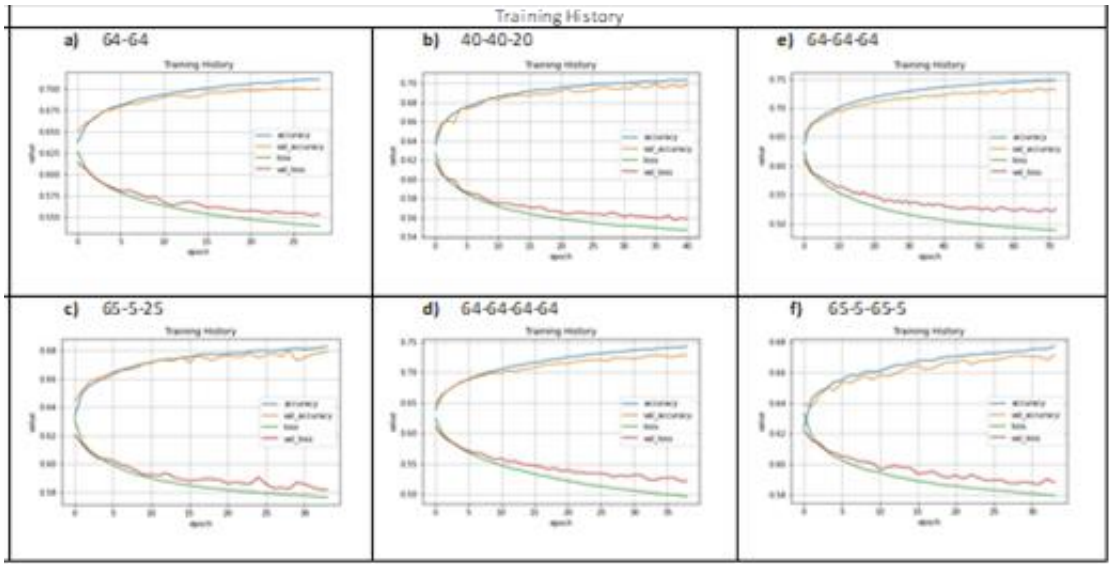
Pada parameter fungsi aktivasi output nilai akuraasi dengan fungsi aktivasi sigmoid sedikit lebih tinggi dibanding softmax pada optimizer Adam, adadelta, dan SGD.

Maka pada model dengan variasi parameter *hidden layer* digunakan parameter optimizer adam dan fungsi aktivasi output sigmoid. Hasil dari model dengan variasi parameter *hidden layer* ditunjukan pada tabel dan gambar.

Tabel 6. Hasil dengan variasi parameter hidden layer pada model ANN

	Data evaluasi	
64-64	loss	0.562404
	categorical_accuracy	0.694271
	val_loss	0.569308
	val_categorical_accuracy	0.689214
40-40-20	loss	0.564871
	categorical_accuracy	0.690659
	val_loss	0.571942
	val_categorical_accuracy	0.686918
64-64-64	loss	0.519783
	categorical_accuracy	0.725662
	val_loss	0.540701
	val_categorical_accuracy	0.714654
65-5-25	loss	0.588206
	categorical_accuracy	0.672642
	val_loss	0.592032
	val_categorical_accuracy	0.671058
64-64-64-64	loss	0.532704
	categorical_accuracy	0.715797
	val_loss	0.547032
	val_categorical_accuracy	0.708315
65-5-65-5	loss	0.591983
	categorical_accuracy	0.664713
	val_loss	0.596878
	val_categorical_accuracy	0.661272

Nilai akurasi paling tinggi didapat dengan formasi node hidden layer 64-64-64 , diikuti dengan formasi 64-64-64-64 . Hal ini menunjukan jumlah node yang sama pada tiap layer menghasilkan nilai akurasi yang lebih tinggi di banding denggan jumlah node yang bervariasi.



Gambar 11. Grafik variasi parameter hhidden layer

4.2 KLASIFIKASI DENGAN KNN DAN DECISION TREE

a) KNN

Berikut adalah hasil uji model dengan Parameter *n* dan *p* yang divariasikan untuk mendapatkan hasil akhir dengan parameter yang menghasilkan nilai terbaik.

Tabel 7. Hasil dengan variasi parameter p pada model KNN

	n=5 p=2 (Euclidean distance)	n=5 p=1 (manhattan distance)

cofusion matrix	`from sklearn.metrics import confusion_matrix`
	`cm = confusion_matrix(y_test, y_pred)`
	`cm`
	`array([[65890, 11120],`
	`[13169, 40726]], dtype=int64)`
		`from sklearn.metrics import confusion_matrix`
		`cm = confusion_matrix(y_test, y_pred)`
		`cm`
		`array([[67534, 9750],`
		`[11913, 41708]], dtype=int64)`

classificaton report	`from sklearn.metrics import classification_report`
	`print(classification_report(y_test, y_pred))`
	precision recall f1-score support
	0 0.83 0.86 0.84 77010
	1 0.79 0.76 0.77 53895
	accuracy 0.81 130905
	macro avg 0.81 0.81 0.81 130905
	weighted avg 0.81 0.81 0.81 130905
		`from sklearn.metrics import classification_report`
		`print(classification_report(y_test, y_pred))`
		precision recall f1-score support
		0 0.85 0.87 0.86 77284
		1 0.81 0.78 0.79 53621
		accuracy 0.83 130905
		macro avg 0.83 0.83 0.83 130905
		weighted avg 0.83 0.83 0.83 130905

Dari tabel menunjukan semakin tinggi nilai TP dan TN semakin baik pula tingkat klasifikasi dari akurasi, presisi, *recall* dan F1-score. Didapatkan nilai precision lebih tinggi yaitu dengan p=1 dibanding p=2. penggunaan eulidien distance lebih baik dari manhattan distance.

Berdasarkan matriks hasil prediksi pada p=1, didapatkan hasil memprediksikan secara benar yaitu sebanyak 67534 dengan selisih kesalahan 9750 dan yang sesuai prediksi salah sebanyak 41708 dengan selisih kesalahan yaitu sebanyak 11913. Selanjutnya pada model dengan variasi parameter *n* digunakan parameter tetap p= 1. Hasil dari model dengan variasi parameter *n* ditunjukan pada tabel .

Dari tabel didapatkan nilai precision lebih tinggi yaitu dengan n=3 dibanding n=5 dan n=7.

Berdasarkan matriks hasil prediksi pada n=3, didapatkan hasil memprediksikan secara benar yaitu sebanyak 70161 dengan selisih kesalahan 7019 dan yang sesuai prediksi salah sebanyak 45748 dengan selisih kesalahan yaitu sebanyak 7977.

b) Decision Tree

Pada tabel adalah hasil uji model dengan parameter *test_size* yang divariasikan untuk mendapatkan hasil klasifikasi dengan parameter yang menghasilkan nilai terbaik pada model *decision tree*.

Dari tabel didapatkan nilai precision lebih tinggi yaitu dengan test_size 0.05 dibanding 0.25 dan 0.5. parameter test_size yang lebih kecil mempengaruhi hasil akurasi dan presisi yang lebih baik sementara dengan memperbesar nilai test_size justru menghasilkan nilai akurasi yang menurun.

Berdasarkan matriks hasil prediksi pada test_size 0.05 didapatkan hasil memprediksikan secara benar yaitu sebanyak 15367 dengan selisih kesalahan 187 dan yang sesuai prediksi salah sebanyak 10437 dengan selisih kesalahan yaitu sebanyak 190.

Tabel 8. Hasil dengan variasi parameter n pada model KNN

	n=3	n=5	n=7

cofusion matrix	`from sklearn.metrics import confusion_matrix`
	`cm = confusion_matrix(y_test, y_pred)`
	`cm`
	`array([[70161, 7019],`
	`[ 7977, 45748]], dtype=int64)`
		`from sklearn.metrics import confusion_matrix`
		`cm = confusion_matrix(y_test, y_pred)`
		`cm`
		`array([[67534, 9750],`
		`[11913, 41708]], dtype=int64)`
			`from sklearn.metrics import confusion_matrix`
			`cm = confusion_matrix(y_test, y_pred)`
			`cm`
			`array([[66245, 11153],`
			`[14619, 38888]], dtype=int64)`

classificaton report	`from sklearn.metrics import classification_report`
	`print(classification_report(y_test, y_pred))`
	precision recall f1-score support
	0 0.90 0.91 0.90 77180
	1 0.87 0.85 0.86 53725
	accuracy 0.89 130905
	macro avg 0.88 0.88 0.88 130905
	weighted avg 0.89 0.89 0.89 130905
		`from sklearn.metrics import classification_report`
		`print(classification_report(y_test, y_pred))`
		precision recall f1-score support
		0 0.85 0.87 0.86 77284
		1 0.81 0.78 0.79 53621
		accuracy 0.83 130905
		macro avg 0.83 0.83 0.83 130905
		weighted avg 0.83 0.83 0.83 130905
			`from sklearn.metrics import classification_report`
			`print(classification_report(y_test, y_pred))`
			precision recall f1-score support
			0 0.82 0.86 0.84 77398
			1 0.78 0.73 0.75 53507
			accuracy 0.80 130905
			macro avg 0.80 0.79 0.79 130905
			weighted avg 0.80 0.80 0.80 130905

Tabel 9. Hasil dengan variasi parameter test_size pada model decission tree

	test_size = 0.05	test_size = 0.25	test_size = 0.5

cofusion matrix	`from sklearn.metrics import confusion_matrix`
	`y_pred = clf.predict(X_test)`
	`confusion_matrix(y_test, y_pred)`
	`array([[15367, 187],`
	`[ 190, 10437]], dtype=int64)`
		`from sklearn.metrics import confusion_matrix`
		`y_pred = clf.predict(X_test)`
		`confusion_matrix(y_test, y_pred)`
		`array([[75409, 1733],`
		`[ 1873, 51890]], dtype=int64)`
			`from sklearn.metrics import confusion_matrix`
			`y_pred = clf.predict(X_test)`
			`confusion_matrix(y_test, y_pred)`
			`array([[146482, 7910],`
			`[ 7875, 99542]], dtype=int64)`

classificaton report	`from sklearn.metrics import classification_report`
	`print(classification_report(y_test, y_pred))`
	precision recall f1-score support
	0 0.99 0.99 0.99 15554
	1 0.98 0.98 0.98 10627
	accuracy 0.99 26181
	macro avg 0.99 0.99 0.99 26181
	weighted avg 0.99 0.99 0.99 26181
		`from sklearn.metrics import classification_report`
		`print(classification_report(y_test, y_pred))`
		precision recall f1-score support
		0 0.98 0.98 0.98 77142
		1 0.97 0.97 0.97 53763
		accuracy 0.97 130905
		macro avg 0.97 0.97 0.97 130905
		weighted avg 0.97 0.97 0.97 130905
			`from sklearn.metrics import classification_report`
			`print(classification_report(y_test, y_pred))`
			precision recall f1-score support
			0 0.95 0.95 0.95 154392
			1 0.93 0.93 0.93 107417
			accuracy 0.94 261809
			macro avg 0.94 0.94 0.94 261809
			weighted avg 0.94 0.94 0.94 261809

5. KESIMPULAN

Machine learning terus dikembangkan dalam penelitian di bidang analisis keylogger. Telah dilakukan membangun sistem deteksi keylogger dan klasifikasi dengan menyajikan metode dengan parameter tertentu yang direkomendasikan berbasis machine learning. Hasil akhir didapat parameter yang menghasilkan nilai terbaik yang ditunjukan dengan nilai accuracy, precision, recall, dan F1-score yang tinggi. Pada sistem deteksi keylogger dengan metode ANN didapat parameter terbaik yaitu fungsi aktivasi output sigmoid, optimizer adam dan hidden layer 64-64-64 dengan nilai akurasi 71%.

Akurasi 71% tidak terlalu tinggi, tetapi model tidak hanya memprediksi kelas mayoritas saja (jika tidak, kita akan memiliki akurasi= 308813/523617~=58,98%). Parameter optimizer cukup mempengaruhi hasil akurasi, sementara parameter fungsi aktivasi dan hidden layer yang dilakukan tidak cukup berpengaruh terhadap nilai akurasi yang didapat. Model ini dapat bekerja lebih lanjut dengan menambahkan lapisan dan neuron dengan jumlah yang lebih besar, mengubah fungsi aktivasi yang lain , parameter pengoptimal, dll.

Pada sistem klasifikasi keylogger didapat parameter terbaik yaitu pada metode KNN parameter manhattan distance (p=1) dan n =3 dengan nilai akurasi 89%. Sementara pada metode Decision Tree dengan parameter test_size 0.05 didapatkan nilai akurasi 99%. Pada model KNN parameter n yaitu jumlah neuron tetangga yang terus ditambah (lebih banyak) menghasilkan nilai akurasi yang menurun. Sementara pada metode Decision Tree parameter test_size yang lebih kecil mempengaruhi hasil akurasi dan presisi yang lebih baik sementara dengan memperbesar nilai test_size justru menghasilkan nilai akurasi yang menurun. Model KNN dan Decision tree ini dapat bekerja lebih lanjut dengan menguji parameter lain atau memvariasikan parameter dengan nilai yang lebih bergam.

REFERENSI

[1] https://www.malwarebytes.com/keylogger

[2] https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[3] Artificial Neural Network — https://www.tutorialspoint.com/artificial_int

elligence/artificial_intelligence_neural_networks.htm

[4] https://keras.io/guides/sequential_model/

[5] Siang, J. J. (2005). Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab. Yogyakarta: Penerbit Andi.

[6] Kusrini, & Luthfi, E. T. (2009). Algoritma Data Mining. Yogyakarta: C.V ANDI OFFSET (Penerbit ANDI).

[7] Gorunescu, F. (2011). Data Mining Concept, Models and Techniques. Berlin Heidelberg: Springer.

[8] https://www.kaggle.com/subhajournal/keylogger-detection