DETEKSI *KAGGLE BOT ACCOUNT* MENGGUNAKAN *DEEP NEURAL NETWORKS*

Virda Virdausih Putri[1)], Abu Tholib[2], Cahyuni Novia[3]
[1, 2,3)]Informatika, Universitas Nurul Jadid
Pondok Pesantren Nurul Jadid Karanganganyar Paiton Probolinggo 67291
e-mail: virdavirdausiahputri@gmail.com[1)], ebuenje@gmail.com[2)], vhie771108@gmail.com[3)]

ABSTRAK

Pengumpulan data dalam penelitian memiliki tantangan yang sulit, terutama di Kaggle, platform populer untuk data scientist. Namun, dalam beberapa tahun terakhir, telah ada banyak laporan tentang adanya indikasi akun palsu di Kaggle yang sulit terdeteksi, sehingga dapat mengancam integritas data dan kredibilitas penelitian. Salah satu ciri utama untuk mengidentifikasi akun palsu adalah dengan melihat profil yang tidak lengkap atau tidak konsisten. Penelitian ini bertujuan membantu pengguna datascience untuk mendeteksi akun bot Kaggle dengan membangun suatu model menggunakan Deep Neural Networks. DNN adalah algoritma machine learning yang meniru sistem syaraf dalam otak manusia. DNN terdiri dari lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. DNN memiliki keunggulan dalam mempelajari pola-pola dari data kompleks dan memberikan hasil yang lebih akurat daripada algoritma Machine Learning tradisional. Penelitian ini menggunakan dataset yang terdiri dari 1.048.574 baris row data dan 17 variabel yang diperoleh dari kaggle.com. Data tersebut kemudian melalui proses pre-processing untuk mempersiapkan dataset dan membangun model DNN dengan 5 hidden layer. Model DNN ini, akan dilatih menggunakan data training dan diuji menggunakan data testing. Hasil penelitian menunjukkan akurasi tinggi, dengan 99,90% akurasi pada data training, 98,42% validasi akurasi, dan 99,86% akurasi pada data testing. Hasil tersebut membuktikan bahwa DNN dapat bekerja secara efektif untuk mendeteksi akun palsu Kaggle. Dengan adanya deteksi akun palsu ini, kualitas penelitian dapat lebih terjamin.

***Kata Kunci**: Deteksi, Kaggle, Bot Account, Machine Learning, Deep Neural Networks*

ABSTRACT

Data collection in research is challenging, especially on Kaggle, a popular platform for data scientists. However, in recent years, there have been many reports of fake accounts on Kaggle that are difficult to detect, threatening data integrity and research credibility. One of the key traits to identify fake accounts is by looking at incomplete or inconsistent profiles. This research aims to help datascience users to detect Kaggle bot accounts by building a model using Deep Neural Networks. DNN is a machine learning algorithm that mimics the nervous system in the human brain. DNN consists of input layers, hidden layers, and output layers. DNN has the advantage of learning patterns from complex data and providing more accurate results than traditional Machine Learning algorithms. This research uses a dataset consisting of 1,048,574 rows of row data and 17 variables obtained from kaggle.com. The data is then pre-processed to prepare the dataset and build a DNN model with 5 hidden layers. This DNN model will be trained using training data and tested using testing data. The results show high accuracy, with 99.90% accuracy on training data, 98.42% validation accuracy, and 99.86% accuracy on testing data. These results prove that DNN can work effectively to detect fake Kaggle accounts. With this fake account detection, the quality of research can be further improved.

***Keywords**: Detection, Kaggle, Bot Account, Machine Learning, Deep Neural Networks*

I. PENDAHULUAN

Dalam suatu penelitian, data memiliki peran penting karena menjadi sumber informasi dan angka yang digunakan untuk memberikan pemahaman terperinci tentang objek penelitian dan mengungkapkan fenomena yang kini terjadi [1] [2] [3]. Data dapat diperoleh dari berbagai sumber, baik melalui internet maupun secara langsung dari lokasi penelitian [4]. Namun, proses pengumpulan data seringkali sulit dilakukan dalam sebuah penelitian yang menyebabkan kasus pemalsuan data sering terjadi [5]. Dalam beberapa tahun terakhir, terdapat banyak laporan yang menunjukkan adanya praktik *bot voting* yang menghasilkan data yang dimanipulasi dalam kompetensi ilmu data di *Kaggle*. Pemalsuan data penelitian dapat menyebabkan krisis kredibilitas *Science*, [6] dimana hasil penelitian tidak disajikan secara akurat dan bahkan dapat menyebabkan pembohongan publik jika hasil penelitian tersebut dipublikasikan.

Kaggle adalah salah satu situs populer dalam bidang *Global Data Science* dan *Machine Learning*, yang memiliki lebih dari 6000 dataset dan komunitas ilmuan terbesar di era ini [7]. Banyak data *scientist*, baik yang baru memulai maupun yang sudah berpengalaman,

aktif menggunakan *Kaggle* karena situs ini menyediakan banyak dataset dan lingkungan *science* berbasis web yang menggunakan teknologi kontainerisasi terkini [8]. Namun, pesatnya perkembangan *Kaggle* dengan berbagai keunggulan layanannya, terindikasi dimanfaatkan oleh pihak-pihak tidak bertanggung jawab untuk kepentingan pribadi dengan membuat akun pengguna palsu (*bot*). Akun *bot* adalah akun yang identitas penggunanya palsu. Akun palsu mengacu pada individu yang menggunakan media sosial untuk menulis, berpendapat, dan terlibat dalam aktivitas *online* tanpa bermaksud agar orang lain mengetahui identitas pribadi mereka [9]. Fenomena akun palsu ini diatur dalam Undang-undang No. 11 Tahun 2008 Tentang Informasi dan Transaksi Elektronik [10].

Penelitian bertujuan untuk membangun dan menghasilkan model prediksi menggunakan metode *Deep Neural Network untuk* membantu pengguna *datascience* dalam mendeteksi akun palsu di situs *Kaggle*. Dengan menggunakan DNN diharapkan penelitian ini dapat memberikan kontribusi dalam deteksi akun palsu di Kaggle. Penelitian ini memiliki fokus pada deteksi Kaggle bot account dan tidak membahas deteksi akun palsu di situs lainnya. Data yang digunakan menggunakan data dari situs Kaggle dan evaluasi performa model akan dilakukan menggunakan *confusion matrix* untuk membandingan hasil kinerja model DNN dengan penelitian sebelumnya.

Penelitian ini banyak mendapatkan inspirasi dan referensi dari penelitian-penelitian sebelumnya sebagai dasar referensi dan pembandingan. Penelitian [11] melakukan klasifikasi akun palsu terhadap Sosial Media *Online* yang menggunakan Algoritma RNN. Hasil penelitian ini menunjukkan bahwa menggunakan RNN untuk mengklasifikasi akun palsu dapat mencapai akurasi tinggi dengan kerugian rendah dengan nilai rata-rata akurasi diatas 80%. Penelitian [12] memiliki tujuan untuk mendeteksi akun twitter bot dengan menggunakan klasifikasi decition tree. Hasil penelitian menunjukkan performa model yang cukup baik, terbukti dari pengukuran akurasi sebesar 88,84%. Penelitian ketiga [13] melakukan analisis perbandingan terhadap beberapa algoritma *Machine Learning* yaitu *SVM, Naïve Bayes, Random Forest, dan Adaptive Boosting* untuk mendeteksi akun palsu pada Instagram. Hasil penelitian menunjukkan bahwa AdaBoost merupakan algoritma yang memiliki akurasi tertinggi sebesar 92.5%, disusul Random Forest sebesar 91,7 %, SVM sebesar 90.7% dan Naïve Bayes sebesar 83,6%. Mustafa, dkk [14] mengusulkan metode *tweet Similarity Feature* dan *Support vector machine* untuk deteksi buzzer pada twitter menghasilkan akurasi 89%.

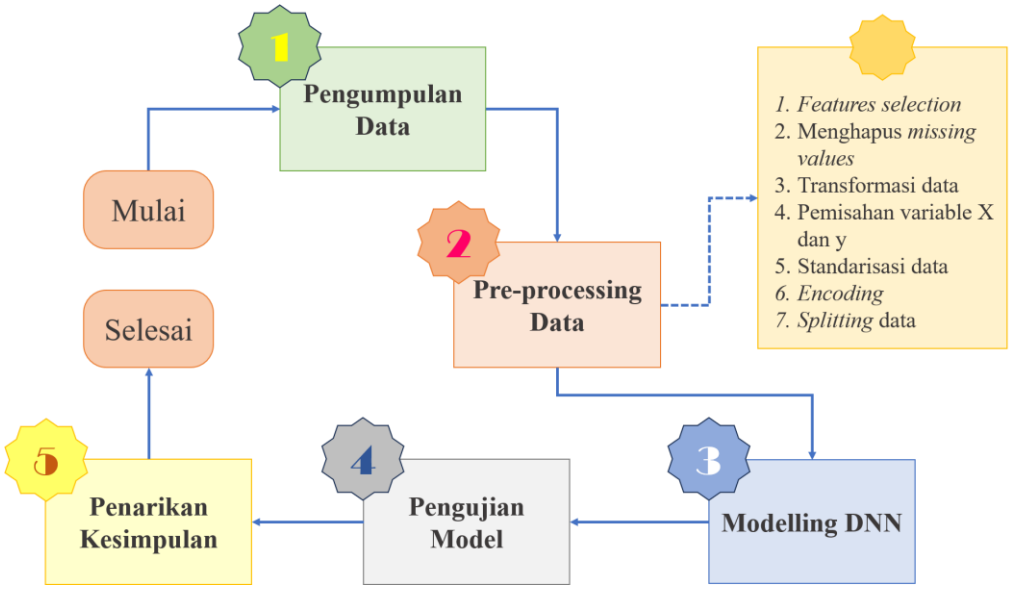
Berdasarkan beberapa referensi penelitian yang telah dilakukan, dapat menunjukkan bahwa penelitian sebelumnya hanya berfokus pada data dan fitur-fitur pada platfrom media sosial umum seperti *Twitter* dan *Instagram*. Namun, hingga saat ini belum ada penelitian khusus yang dilakukan untuk mendeteksi akun palsu pada *Kaggle*. *Kaggle* memiliki karakteristik dan fitur yang berbeda dari platform sosial media lainnya, seperti *dataset*, kompetensi *science*, dan diskusi forum. Oleh karena itu, diperlukan penelitian yang memahami konteks dan karakteristik khusus dari *Kaggle*, dan mengidentifikasi fitur-fitur relevan dalam mendeteksi akun palsu. Selanjutnya, dalam penelitian sebelumnya cenderung menggunakan algoritma *machine learning* tradisional yang membutuhkan waktu *training* lebih lama, maka untuk itu perlu dilakukan penelitian untuk menemukan metode yang lebih canggih dan akurat untuk mendeteksi akun palsu *Kaggle*, seperti menggunakan *Deep Neural Networks.*

Deep Neural Networks (DNN) adalah jenis algoritma pembelajaran mesin yang memiliki kesamaan dengan struktur saraf tiruan dan bertujuan untuk meniru cara pemrosesan informasi yang terjadi pada otak manusia [15]. *DNN* menghasilkan kinerja yang canggih dalam berbagai aplikasi di bidang pembelajaran mesin dan kecerdasan buatan [16]. *Deep Neural Networks* dipilih karena memiliki kemampuan untuk mempelajari pola dan fitur yang kompleks dari data mentah, mampu mengekstraksi fitur-fitur yang relevan dari data mentah secara otomatis. Hal ini sangat bermanfaat dalam mengolah *data Kaggle bot Account* yang memiliki tingkat kompleksitas yang tinggi. Penelitian ini, menggunakan dataset *Kaggle Bot Account* yang berjumlah 1.048.574 baris data dengan 9 *independent variabel* yang terdiri dari jumlah *followers*, jumlah mengikuti akun sesorang, jumlah publikasi dataset, jumlah publikasi *code*, jumlah diskusi yang diikuti, rata-rata waktu yang dihabiskan untuk membaca buku catatan dalam hitungan menit, jumlah total *vote* yang diberikan individu ke kumpulan *notebook*, jumlah total *vote* yang diberikan individu ke kumpulan data, jumlah total *vote* yang diberikan individu untuk komentar. Penelitian ini juga melibatkan satu *dependent variabel* yaitu ISBOT. Pemilihan variabel-variabel tersebut didasarkan pada harapan bahwa variabel-variabel tersebut akan memberikan informasi relevan dan bermanfaat dalam deteksi akun *bot Kaggle*, yang telah dibuktikan melalui analisis *correlation matrix.*

Deteksi *Kaggle bot Account* menggunakan metode DNN menjadi sangat penting untuk menjaga integritas dan kredibilitas data dan kompetensi di situs *Kaggle*. Hal ini mendesak untuk mengidentifikasi dan menghapus akun *bot* yang mencurigakan serta memastikan bahwa dataset yang diunggah asli. Selain itu, langkah-langkah untuk mengatasi akun *bot* juga akan melindungi kepentingan dan reputasi pengguna yang berdidikasi dan berusaha meningkatkan keterampilan dalam analisis data.

II. METODE PENELITIAN

Metode penelitian dapat memberikan gambaran dari tahapan penelitian yang akan dilakukan. Dengan adanya metode penelitian diharapkan dapat mempermudah proses penelitian dalam mendeteksi *Kaggle Bot Account* menggunakan *Deep Neural Networks*. Berdasarkan Gambar 1, metode penelitian ini dimulai dengan tahap pengumpulan data. Setelah itu, data mentah akan melalui proses *pre-processing* untuk mempersiapkan data agar lebih mudah digunakan pada tahap pemodelan. Selanjutnya, dilakukan pembangunan model *Deep Neural Networks*. Setelah model dibangun, diperlukan pengujian model untuk mengevaluasi kinerja model DNN. Terakhir, dilakukan penarikan kesimpulan berdasarkan hasil pengujian tersebut.



Gambar 1. Metode Penelitian

A. Pengumpulan Data

Dataset yang digunakan dalam penelitian ini merupakan dataset yang tersedia secara publik dan diperoleh dari situs kaggle.com. Dataset tersebut berjumlah 1.048.574 data *record* dengan 16 variabel atribut dan 1 variabel target.

B. *Pre-processing Data*

Data *Kaggle Bot Account* yang didapat masih cukup berantakan hingga memerlukan tahap *pre-processing* data. Tahap *pre-processing* dalam penelitian ini meliputi *features selection*, menghapus *missing values*, *data transformation*, pemisahan data variabel X dan y, standarisasi data, *encoding* serta *splitting data* menjadi data *training* sebesar 80% dan *data testing* sebesar 20%. Hal ini dilakukan untuk menyiapkan data agar dapat melanjutkan ke langkah-langkah selanjutnya dengan lebih mudah dan terstruktur [17].

C. *Modelling Deep Neural Networks*

Deep Neural Networks (DNN) adalah algoritma pembelajaran mesin yang memiliki kesamaan dengan jaringan saraf tiruan dan memiliki tujuan untuk meniru pemrosesan informasi yang terjadi dalam otak manusia [1]. DNN Terdiri dari 3 lapisan yaitu lapisan masukan, lapisan tersembunyi dan lapisan keluaran [18]. Jika jumlah lapisan tersembunyi lebih besar dari atau sama dengan tiga, maka sistem dilambangkan sebagai DNN [19]. Beberapa peneliti merekomendasikan penggunaan model prediksi DNN untuk mencapai tingkat akurasi yang lebih tinggi dan menghindari masalah *underfitting* dan *overfitting* [20]. *Deep Neural Networks* pada persamaan (1) menunjukan variabel *input* memiliki nilai bobot (w) yang akan melakukan operasi linear dengan menjumlahkan semua nilai bobot variabel *input* ∑(WnXn) dan ditambah nilai bias (b). Kemudian, hasilnya (z) akan dioperasikan sebagai fungsi aktivasi (σ).

$$y_i = \sigma\{\sum_i^n w^i x^i + b\} \quad (1)$$

Model DNN memiliki 5 *hidden layer* dengan fungsi aktivasi *Relu. Rectifier Linier Unit* (ReLU) adalah fungsi aktivasi yang memiliki perhitungan sederhana [21]. Dengan ReLU elemen input yang bernilai kurang dari nol maka nilainya akan dibatasi menjadi 0, jika lebih dari 0 maka menjadi fungsi linier. ReLu dapat dilihat pada persamaan (2).

$$\sigma = \max(0, z) = \begin{cases} z_i, if\ z_i \geq 0 \\ z,, if\ z_i < 0 \end{cases} \quad (2)$$

Fungsi aktivasi *softmax* diterapkan pada *output layer* karena merupakan salah satu fungsi aktivasi untuk menangani masalah *multi-class*. Berdasarkan persamaan (3) nilai *neuron* dari lapisan *output* (z) akan melalui perhitungan eksponensial (e) sebagai fungsi *non-linear*. Kemudian nilai-nilai tersebut akan dibagi dengan jumlah semua nilai eksponensial dari variabel *input* ∑jexp(xj) untuk dinormalisasi dan menjadi nilai probabilitas. Probabilitas tertinggi mengindikasikan kelas terpilih [22]. Turunan fungsi aktivasi *softmax* dapat terlihat pada persamaan (4).

$$\sigma(z)j = \frac{e^z j}{\sum_{k=1}^{K} e^z k} \quad (3)$$

$$\frac{\partial p_i}{\partial x_j} = p_i(\delta_{ij} - p_j)\ dengan\ \delta_{ij} = \begin{cases} 1, jika\ i = j \\ 0, jika\ i \neq j \end{cases} \quad (4)$$

D. Pengujian Model

Tahapan pengujian model dilakukan untuk mengukur tingkat keberhasilan model yang telah dibuat [23]. Proses pengujian model dilakukan dengan melakukan pengujian terhadap model *Deep Neural Networks* menggunakan metode *Confusion Matrix* yang menampilkan nilai *precision, recall, f1-score,* dan *support.* Pengujian model DNN menggunakan *tools Google Collab* dengan bahasa pemrograman *Python*.

E. Penarikan Kesimpulan

Setelah melakukan seluruh tahapan penelitian diatas, pada tahap ini akan dilakukan analisis terhadap uji coba model DNN yang telah dibuat. Sehingga dapat disimpulkan implementasi DNN dalam deteksi *Kaggle Bot Account* dapat bekerja dengan baik atau sebaliknya.

III. HASIL DAN PEMBAHASAN

A. Pengumpulan Data

Dataset berupa data *Kaggle Bot Acoount* yang didapatkan dari situs kaggle.com dengan total 1.048.574 data *record* yang dilengkapi dengan 16 *independent variabel* dan 1 *dependent variabel*. Berikut sampel awal dataset yang belum dikelola, ditunjukkan pada Gambar 2.

unnamed: 0	NAME	GENDER	EMAIL_ID	IS_GLOGIN	FOLLOWER_COUNT	FOLLOWING_COUNT	DATASET_COUNT	...	ISBOT
0	Johnny Ke	Male	jacksonala	FALSE	53.0	87.0	5.0	...	
1	Dwayne L	Male	calvin80@	TRUE	16.0	67.0	5.0	...	
2		Male	qbrown@	TRUE	44.0	81.0	4.0	...	FALSE
3	Russell Si	Male	kimberlyv	TRUE	23.0	114.0	5.0	...	FALSE
4	Jamie Wil	Female	shaunbroc	FALSE	46.0	112.0	2.0	...	FALSE
5	Elijah Park	Male	mpearson	FALSE	2.0	2.0	0.0	...	TRUE
6	Logan Zim	Male	sparkschristy@exam		46.0	36.0	0.0	...	FALSE
7	Erin Herre	Female		FALSE	2.0	1.0	0.0	...	TRUE
8	Matthew I	Male	harrisregi	TRUE	50.0	25.0	1.0	...	FALSE
9	Michael S	Male	klopez@e	TRUE	65.0	99.0	7.0	...	FALSE
10	Gregory G	Male	todd84@e	FALSE		44.0	4.0	...	
...	...	...	...	...	...	...	...	...	...
1048574	Christoph	Male	traci31@e	FALSE	1.0	2.0	0.0	...	TRUE

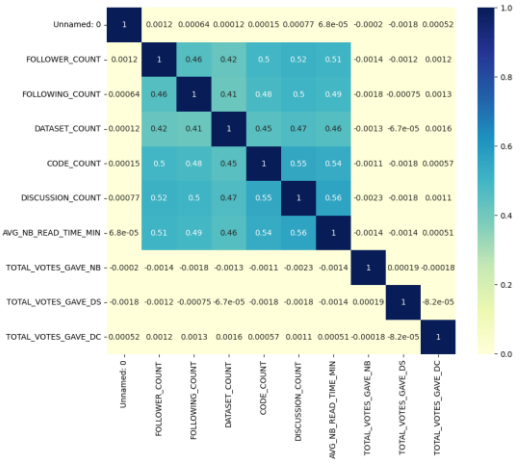
Gambar 2. Sampel Data Awal

B. *Pre-processing Data*

Tahap *pre-processing* data merupakan langkah penting sebelum menggunakan data untuk membangun model prediksi [19]. Tujuan dari proses *pre-processing* adalah untuk memastikan dataset tidak mengandung data ambigu atau data kosong yang dapat menurunkan kualitas *dataset* [16].

1) *Features Selection* dan *Missing Values*

Pada tahap ini, akan dilakukan proses *fiture selection* terhadap 16 *independent variabel* untuk memilih attribute *importance* dalam dataset. Tindakan ini dilakukan, karena tidak seluruh data dan atribut bisa dipakai [24]. *Features selection* bertujuan untuk memilih fitur optimal serta membuang data tidak relevan dan berlebih, menghindari *overfitting*, dan menyederhanakan model. *Features selection* pada penelitian ini menggunakan korelasi matriks sebagai dasar pemilihan fitur optimal.



Gambar 3. Matriks Korelasi

Metriks korelasi merupakan sebuah tabel yang memperlihatkan tingkat hubungan antara dua atau lebih variabel yang memiliki rentang nilai antara -1 hingga +1 [25]. Dari Gambar 3 dapat dilihat bahwa semakin pekat warna biru, maka semakin tinggi korelasi antar variabel. Hal ini menunjukkan bahwa ada hubungan positif yang kuat antar variabel. Disisi lain, warna putih menunjukkan hubungan negatif antar variabel, yang artinya tingkat korelasinya rendah.

Berdasarkan hasil matriks korelasi hanya terdapat 6 *independent variabel* yang dianggap optimal. Variabel-variabel *independent variabel* tersebut adalah *FOLLOWER_COUNT, FOLLOWING_COUNT, DATASET_COUNT, CODE_COUNT, DISCUSSION_COUNT, AVG_NB_READ_TIME_MIN*. Variabel *TOTAL_VOTES_GAVE_NB, TOTAL_VOTES_GAVE_DS, TOTAL_VOTES_GAVE_DC* tetap digunakan dipilih meskipun memiliki tingkat korelasi yang rendah karena untuk memperkaya data. Total *independent variabel* yang digunakan sebanyak 9 variabel. Variabel unnamed: 0 tidak dipilih karena merupakan no urut data. Untuk satu variabel *dependent* "ISBOT" terdiri dari 2 kelas yaitu *True* dan *False.* Variabel yang digunakan dijelaskan pada Tabel 1.

TABEL 1.
VARIABEL TERPILIH

Variabel	Penjelasan
FOLLOWER_COUNT	Jumlah pengikut yang dimiliki individu tersebut
FOLLOWING_COUNT	Jumlah individu yang diikuti oleh individu tersebut
DATASET_COUNT	Jumlah set data yang telah dibuat oleh individu
CODE_COUNT	Jumlah notebooks yang telah dibuat oleh individu
DISCUSSION_COUNT	Jumlah diskusi yang telah diikuti oleh individu tersebut
AVG_NB_READ_TIME_MIN	Waktu rata-rata yang dihabiskan untuk membaca buku catatan setiap menit
TOTAL_VOTES_GAVE_NB	Jumlah total suara yang diberikan individu kepada notebook
TOTAL_VOTES_GAVE_DS	Jumlah total suara yang diberikan individu ke set data
TOTAL_VOTES_GAVE_DC	Jumlah total suara yang diberikan individu untuk komentar diskusi
ISBOT	Variabel target yang berisi 2 label (*True/False*) yang menunjukkan apakah individu tersebut adalah *bot* atau bukan

Dataset pada penelitian ini masih memiliki *missing values* dengan jumlah yang cukup banyak seperti pada Gambar 4. Maka untuk itu menghapus baris yang mengandung *missing values* merupakan salah satu cara untuk mengatasinya hingga data menjadi lebih utuh dan model mudah membaca data. Hasil dari proses *features selection* dan *missing values* dapat dilihat pada Gambar 5.

```
Data columns (total 17 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   Unnamed: 0            1321188 non-null  int64
 1   NAME                  1243024 non-null  object
 2   GENDER                1243309 non-null  object
 3   EMAIL_ID              1243374 non-null  object
 4   IS_GLOGIN             1243272 non-null  object
 5   FOLLOWER_COUNT        1243476 non-null  float64
 6   FOLLOWING_COUNT       1242743 non-null  float64
 7   DATASET_COUNT         1242621 non-null  float64
 8   CODE_COUNT            1243262 non-null  float64
 9   DISCUSSION_COUNT      1243466 non-null  float64
 10  AVG_NB_READ_TIME_MIN  1242872 non-null  float64
 11  REGISTRATION_IPV4     1242859 non-null  object
 12  REGISTRATION_LOCATION 1242898 non-null  object
 13  TOTAL_VOTES_GAVE_NB   1243483 non-null  float64
 14  TOTAL_VOTES_GAVE_DS   1243254 non-null  float64
 15  TOTAL_VOTES_GAVE_DC   1243158 non-null  float64
 16  ISBOT                 1242688 non-null  object
dtypes: float64(9), int64(1), object(7)
memory usage: 171.4+ MB
Unnamed: 0                 0
NAME                   78164
GENDER                 77879
EMAIL_ID               77814
IS_GLOGIN              77916
FOLLOWER_COUNT         77712
FOLLOWING_COUNT        78445
DATASET_COUNT          78567
CODE_COUNT             77926
DISCUSSION_COUNT       77722
AVG_NB_READ_TIME_MIN   78316
REGISTRATION_IPV4      78329
REGISTRATION_LOCATION  78290
TOTAL_VOTES_GAVE_NB    77705
TOTAL_VOTES_GAVE_DS    77934
TOTAL_VOTES_GAVE_DC    78030
ISBOT                  78500
dtype: int64
```

Gambar 4. Data sebelum *Features Selection* dan *Missing Values*

```
Data columns (total 10 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   FOLLOWER_COUNT        718852 non-null  float64
 1   FOLLOWING_COUNT       718852 non-null  float64
 2   DATASET_COUNT         718852 non-null  float64
 3   CODE_COUNT            718852 non-null  float64
 4   DISCUSSION_COUNT      718852 non-null  float64
 5   AVG_NB_READ_TIME_MIN  718852 non-null  float64
 6   TOTAL_VOTES_GAVE_NB   718852 non-null  float64
 7   TOTAL_VOTES_GAVE_DS   718852 non-null  float64
 8   TOTAL_VOTES_GAVE_DC   718852 non-null  float64
 9   ISBOT                 718852 non-null  object
dtypes: float64(9), object(1)
memory usage: 60.3+ MB
FOLLOWER_COUNT         0
FOLLOWING_COUNT        0
DATASET_COUNT          0
CODE_COUNT             0
DISCUSSION_COUNT       0
AVG_NB_READ_TIME_MIN   0
TOTAL_VOTES_GAVE_NB    0
TOTAL_VOTES_GAVE_DS    0
TOTAL_VOTES_GAVE_DC    0
ISBOT                  0
dtype: int64
```

Gambar 5. Data setelah *Features Selection* dan *Missing Values*

2) *Data Transformation*

Data variabel y, yang merupakan data target masih mengandung nilai-nilai berbentuk teks yang menyebabkan model tidak dapat membaca data secara langsung. Untuk itu maka dilakukan *conversi type* data variabel "ISBOT" yang awalnya memiliki tipe data object menjadi integer. Ketika tipe data diganti maka label *False* berubah menjadi 0 dan *True* menjadi 1, seperti yang telihat pada Gambar 6.

ISBOT
0
0
0
1
1

Gambar 6. Data Variabel y setelah Konversi Data

3) Pemisahan Data *Input* X dan Data *Output* y

Data *input* X merupakan data latih yang menampung data dari 9 kolom *independent variabel*, seperti yang terlihat pada Gambar 7. Sedangkan data *output* y menampung data dari satu *dependent variabel* yang berisi dua label (*True/False*). Data *output* y terlihat pada Gambar 8.

FOLLOWER_COUNT	FOLLOWING_COUNT	DATASET_COUNT	CODE_COUNT	DISCUSSION_COUNT	AVG_NB_READ_TIME_MIN	TOTAL_VOTES_GAVE_NB	TOTAL_VOTES_GAVE_DS	TOTAL_VOTES_GAVE_DC
44.0	81.0	4.0	17.0	125.0	7.75	16.0	4.0	0.0
23.0	114.0	5.0	24.0	67.0	13.40	21.0	10.0	1.0
46.0	112.0	2.0	12.0	63.0	24.83	10.0	6.0	2.0
2.0	2.0	0.0	0.0	0.0	0.62	18.0	9.0	2.0

Gambar 7. Data *Input* X

	ISBOT
2	False
3	False
4	False
5	True

Gambar 8. Data *Output* y

4) Standarisasi Data

Teknik standarisasi data yang dilakukan adalah dengan melakukan perubahan skala dimana data akan diubah sehingga memiliki rentang rata-rata = 0 dan standart deviasi = 1. Dengan adanya standarisasi data maka data akan menjadi seragam dan model lebih mudah menerima masukan dengan baik. Data yang dinormalisasi merupakan data variabel X. Adapun hasil normalisasi dapat ditemukan pada Gambar 9.

```
array([ 0.74037432,  0.91092038,  0.57325211,  0.80105032,  1.24337276,
       -0.52215681, -0.32674739, -1.09120277, -1.3408341 ])
```

Gambar 9. Data Standarisasi

5) *Encoding*

Metode Encoding yang digunakan adalah dengan *OneHotEncoding*. Metode ini bertujuan untuk mengubah integer variabel y menjadi nilai biner sehingga komputer akan mudah memproses data. Data hasil *encoding* dapat dilihat pada Gambar 10.

```
[[1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]]
```

Gambar 10. Variabel y setelah *Encoding*

6) *Splitting Data*

Pada tahap ini, dataset akan dibagi menjadi 2 bagian sebelum proses *training data*. Metode pembagian dilakukan dengan membagi keseluruhan *dataset* menjadi 80% untuk *training data* dan 20% untuk testing data. Pembagian data tersebut menghasilkan 575.081 baris data *training* dan 143.771 baris data *testing*. Selain itu, pada proses ini data *training* akan diacak sebanyak 90.000x untuk memastikan konsistensi hasil *training*.

C. *Modelling Data*

Pada tahap ini dilakukan pembuatan model menggunakan Algoritma *Deep Neural Networks*. Model ini menggunakan total 718.852 *data record* yang dibagi menjadi 575.081 data *training* dan 143.771 data *testing* untuk melatih data dengan model DNN. Model DNN dibangun menggunakan model *sequential keras API* seperti pada Gambar 11.

```
model = Sequential() #Model
model.add(InputLayer(input_shape=(9,))) #Input Layer
model.add(Dropout(0.3))
model.add(Dense(9, activation='relu')) #Hidden Layer 1
model.add(Dropout(0.3))
model.add(Dense(9, activation='relu')) #Hidden Layer 2
model.add(Dropout(0.3))
model.add(Dense(9, activation='relu')) #Hidden Layer 3
model.add(Dropout(0.3))
model.add(Dense(9, activation='relu')) #Hidden Layer 4
model.add(Dropout(0.3))
model.add(Dense(9, activation='relu')) #Hidden Layer 5
model.add(Dropout(0.3))
model.add(Dense(2, activation='softmax')) #Output Layer
```

Gambar 11. *DNN Model with Keras*

Berdasarkan Gambar 11, dapat diketahui bahwa model DNN memiliki lapisan *input* yang terdiri dari 9 neuron yang merupakan variabel *independent* dari dataset. *Input layer* melakukan *dropout* sebesar 30% untuk mencegah *overfitting* pada model dengan melakukan menghentian secara acak.

Pada lapisan selanjutnya menggunakan 5 *hidden layer* masing-masing terdiri dari 9 *neuron*. Terdapat juga *dropout* sebesar 30% yang diterapkan pada setiap *hidden layer*. Fungsi aktivasi yang digunakan pada *hidden layer* adalan fungsi aktivasi *relu*. *Output layer* menggunakan lapisan dengan 2 *neuron* yang memiliki fungsi aktivasi *softmax* karena kasus penelitian ini termasuk kasus *multi-class/binary-class.*

Categorical-crossentropy digunakan untuk menghitung nilai kerugian karena fungsi ini merupakan perhitungan kerugian untuk fungsi aktivasi *softmax* pada

output layer. Untuk melakukan optimasi model, digunakan metode *adam* yang merupakan parameter default dari keras. Sedangkan, untuk evaluasi model menggunakan *metrik accuracy*. Hasil model *deep neural networks* dengan jelas dapat dilihat pada Gambar 12.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dropout (Dropout)           (None, 9)                 0

 dense (Dense)               (None, 9)                 90

 dropout_1 (Dropout)         (None, 9)                 0

 dense_1 (Dense)             (None, 9)                 90

 dropout_2 (Dropout)         (None, 9)                 0

 dense_2 (Dense)             (None, 9)                 90

 dropout_3 (Dropout)         (None, 9)                 0

 dense_3 (Dense)             (None, 9)                 90

 dropout_4 (Dropout)         (None, 9)                 0

 dense_4 (Dense)             (None, 9)                 90

 dropout_5 (Dropout)         (None, 9)                 0

 dense_5 (Dense)             (None, 2)                 20

=================================================================
Total params: 470
Trainable params: 470
Non-trainable params: 0
_________________________________________________________________
```

Gambar 12. DNN Model

Tahap selanjutnya adalah melakukan proses *training* pada model. Proses *training* model akan melatih data *training*. Sebelum memulai pelatihan, data training akan dibagi menjadi data validasi sebesar 1% data total data. Hal ini dilakukan untuk mengetahui performa model pada data yang tidak dipakai selama proses *training*. Selanjutnya, data akan belajar selama 100x dengan 100 *epoch*. Untuk mempercepat proses pelatihan pada *big* data, data akan dibagi *menjadi batch-batch* dengan ukuran 8.500. Hal ini dilakukan untuk mengoptimalkan waktu yang diperlukan untuk melatih model.

```
Epoch 96/100
61/61 [==============================] - 1s 21ms/step - loss: 0.0492 - accuracy: 0.9838 - val_loss: 0.0030 - val_accuracy: 0.9990
Epoch 97/100
61/61 [==============================] - 1s 21ms/step - loss: 0.0488 - accuracy: 0.9839 - val_loss: 0.0029 - val_accuracy: 0.9990
Epoch 98/100
61/61 [==============================] - 1s 21ms/step - loss: 0.0475 - accuracy: 0.9844 - val_loss: 0.0029 - val_accuracy: 0.9990
Epoch 99/100
61/61 [==============================] - 1s 21ms/step - loss: 0.0486 - accuracy: 0.9842 - val_loss: 0.0027 - val_accuracy: 0.9990
Epoch 100/100
61/61 [==============================] - 2s 26ms/step - loss: 0.0479 - accuracy: 0.9842 - val_loss: 0.0029 - val_accuracy: 0.9990
```

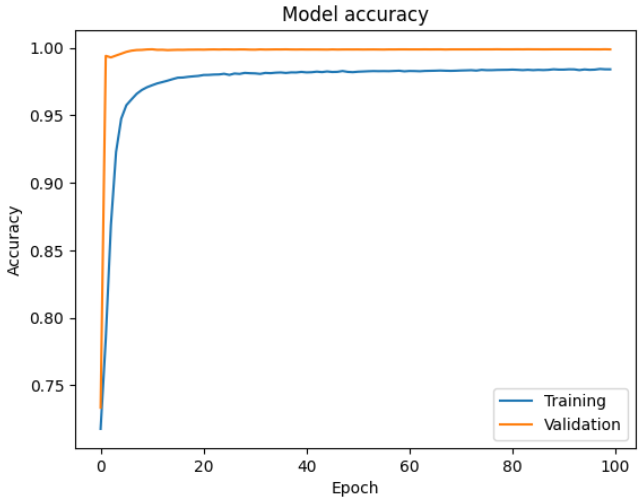
Gambar 13. Proses *Training Model*

Berdasarkan hasil *training* yang terlihat pada Gambar 13, dapat dilihat bahwa model berhasil mencapai tingkat akurasi yang tinggi pada epoch ke-100. Nilai akurasi sebesar 0,9842 menunjukkan bahwa model mampu mengklasifikasi data dengan akurasi 98,42%. Selain itu, nilai *loss* yang diperoleh sebesar 0,0479 menunjukkan bahwa model telah berhasil mengurangi kesalahann prediksi dari nilai sebenarnya. Selanjutnya, pada hasil validasi, terlihat bahwa model mencapai tingkat akurasi yang sangat tinggi, yaitu 0,9990. Hal ini menunjukkan bahwa model juga mampu melakukan prediksi dengan akurasi yang tinggi pada data yang belum pernah dilihat sebelumnya. Selain itu, nilai *loss* validasi yang rendah, yaitu 0,0029, menunjukkan bahwa model memiliki kemampuan yang baik dalam meminimalkan kesalahan prediksi pada data validasi.

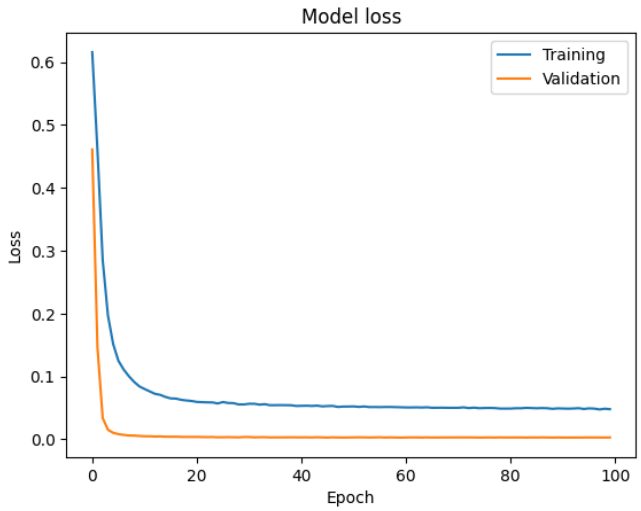
Berdasarkan hasil yang terlihat pada Gambar 14, terdapat tren yang menunjukkan peningkatan akurasi seiring dengan peningkatan jumlah *epoch* pada proses *training* model DNN. Pada *epoch* pertama, nilai akurasi pada data *training* dimulai dari 0,7176, yang menunjukkan bahwa model awalnya memiliki tingkat keakuratan yang relatif rendah. Namun, seiring dengan peningkatan jumlah *epoch*, nilai akurasi secara bertahap meningkat. Selanjutnya, terlihat bahwa nilai akurasi pada data training mengalami fluktuasi, yaitu naik turun. Hal ini umum terjadi pada model. Fluktuasi disebabkan oleh kompleksitas data yang sedang dilatih. Pada *epoch* ke-100, didapatkan nilai akurasi pada data *training* sebesar 0,9842 dan nilai akurasi pada data validasi sebesar 0,9990. Hal ini menunjukkan bahwa model telah mencapai tingkat keakuratan yang sangat tinggi dan mampu memprediksi data *training* dan validasi dengan baik. Meskipun terjadi fluktuasi akurasi selama *training*, secara keseluruhan terlihat adanya tren peningkatan akurasi yang stabil seiring bertambahnya jumlah epoch. Hal ini membuktikan bahwa model DNN berlajar dari data dan meningkatkan kemampuannya dalam melakukan deteksi akun palsu Kaggle dengan akurasi yang lebih tinggi. Tingkat akurasi yang tinggi pada data validasi menunjukkan bahwa model DNN mampu menggeneralisasi data dengan baik dan tidak terjadi *overfitting* pada data *training*.

Berdasarkan hasil yang terlihat pada Gambar 15, terdapat tren yang menunjukkan penurunan nilai *loss* (kerugian) seiring dengan peningkatan jumlah *epoch* pada proses training model DNN. Pada *epoch* pertama, nilai training *loss* dimulai dari 0,6160, sedangkan nilai *validation loss* dimulai dari 0,4611. Hal ini menunjukkan bahwa model awalnya memiliki tingkat kerugian relatif tinggi pada data *training* dan validasi. Namun, seiring peningkatan jumlah *epoch*, terlihat bahwa nilai *loss* secara konsisten menurun dengan stabil. Hal ini menunjukkan bahwa model secara bertahap mengurangi jumlah kesalahan dalam memprediksi data. Pada *epoch* ke-100, didapatkan nilai *training loss* sebesar 0,0479 dan nilai validasi *loss* sebesar 0,0029. Hal ini

menunjukkan bahwa model telah berhasil mengurangi jumlah *loss* dengan sangat baik dan mencapai tingkat kerugian yang sangat rendah. Penurunan nilai loss yang stabil dan konsisten seiring dengan peningkatan jumlah *epoch* menunjukkan bahwa model DNN secara efektif mempelajari pola dalam data dan pengoptimalkan parameternya untuk mengurangi kerugian. Berdasarkan hasil tersebut menunjukkan bahwa model DNN telah berhasil mengurangi jumlah *loss* dengan sangat baik dan mencapai tingkat *loss* yang sangat rendah. Penurunan nilai loss yang stabil dan konsisten seiring meningkatnya jumlah *epoch* menunjukkan bahwa model DNN dapat mempelajari pola data dan mengoptimalkan parameternya untuk mengurangi kerugian. Hasil tersebut menunjukkan bahwa model DNN dapat mengurangi tingkat kesalahan dalam mendeteksi akun *bot kaggle.*



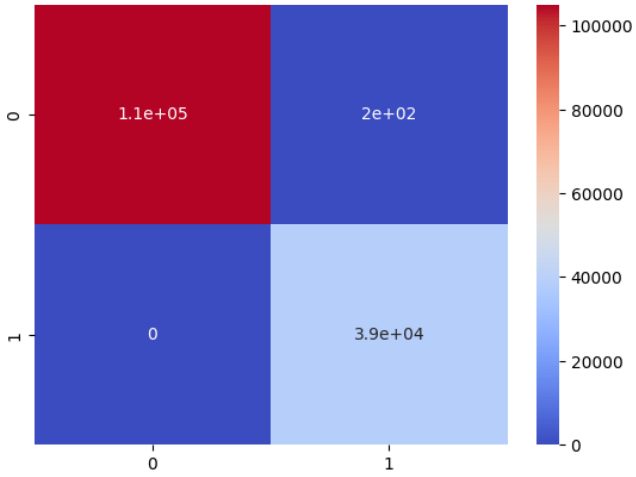
Gambar 14. Grafik *Training dan Validation Accuracy*



Gambar 15. Grafik *Training dan Validation Loss*

D. Pengujian Model

Setelah proses *training* pada model, maka langkah berikutnya adalah melakukan tahap uji coba untuk mengevaluasi kinerja model dengan menggunakan *Confusion Matrix*. Data yang digunakan untuk pengujian sebanyak 20% dari *dataset* dengan total 143.771 data *testing*. Dapat diketahui hasil *Confusion Matrix* pada Gambar 16, bahwa dari 143.771 data *testing* terdapat 200 akun *bot* dan sebanyak 143.571 akun asli dengan nilai akurasi sebesar 0,9986. Hasil dari pengujian *Confusion Matrix* mendapatkan nilai akurasi sebesar 0,9986 atau 99,86%. Nilai tersebut menunjukkan bahwa model DNN memiliki performa yang baik dalam prediksi akun asli namun masih terdapat beberapa kesalahan dalam memprediksi akun palsu.



Gambar 16. *Confusion Matrix Data Testing*

IV. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan pada dataset *Kaggle Bot Account* yang terdiri 9 variabel *independent* dan 1 variabel *dependent* dengan total 718.852 data, diperoleh hasil yang sangat baik dalam deteksi akun palsu *Kaggle*. Model yang dilatih mencapai nilai akurasi *training* 0.9842 dengan *loss* 0,0479, nilai validasi akurasi 0,9990, validasi loss 0,0029, dan nilai akurasi pada data testing sebesar 0,9986. Berdasarkan hasil-hasil ini dapat disimpulkan bahwa model yang dilatih memiliki performa yang sangat baik dalam deteksi akun palsu *Kaggle*. Tingkat akurasi yang tinggi dan nilai loss yang rendah menunjukkan bahwa model dapat mengklasifikasikan akun dengan akurat dan meminimalkan kesalahan mendeteksi *Kaggle Bot Account.*

Kesimpulan yang dapat ditarik hanya didasarkan pada analisis data *Kaggle Bot Account* yang diperoleh. Oleh karena itu, diharapkan adanya penelitian lebih lanjut mengenai deteksi *Kaggle Bot Account* dengan menggunakan metode penelitian yang berbeda, melibatkan sampel yang lebih besar, dan pengembangan implementasi penelitian yang lebih komprehensif dan menyeluruh.

DAFTAR PUSTAKA

[1] S. Siyoto and A. Sodik, *Dasar Metodologi Penelitian*. Yogyakarta: Literasi Media Publishing, 2015.

[2] M. Arhami and M. Nasir, *Data Mining Algoritma dan Implementasi*. Yogyakarta: Penerbit Andi, 2020.

[3] L. Muflikhah, D. Eka Ratnawati, and R. Regasari MP, *Data Mining*. Malang: Universitas Brawijaya Press, 2018.

[4] R. Kurniawan, G. Putra Danu Sohibien, and R. Rahani, *Cara Mudah Belajar Statistik Analisis Data & Eksplorasi*. Jakarta: Prenada Media, 2019.

[5] Suyono, R. Amaliah, D. Ariani, and A. Luciandika, *Cerdas Menulis Karya Ilmiah*. Malang: Gunung Samudera, 2015.

[6] R. A. Sani, *Menulis Laporan Penelitian dan Artikel Ilmiah*. Mojokerto: Komunitas Penulis Kreatif, 2022.

[7] Z. M. Chng, C. Daniel, C. Stefania, and A. Tam, *Python for Machine Learning*. Inggris: Machine Learning Mastery, 2022.

[8] L. Quaranta, F. Calefato, and F. Lanubile, "KGTorrent: A Dataset of Python Jupyter Notebooks from Kaggle," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, Madrid, Spain: IEEE, May 2021, pp. 550–554. doi: 10.1109/MSR52588.2021.00072.

[9] A. R. Maizuly, B. Hartono, and I. Satria, "Penerapan Sanksi Pidana Terhadap Pelaku Tindak Pidana Manipulasi dan Penciptaan melalui Akun Media Sosial Facebook," *JIC*, vol. 6, no. 1, p. 12, Apr. 2022, doi: 10.35308/jic.v6i1.3794.

[10] P. R. Indonesia, "Undang-undang Republik Indonesia Nomor 11 Tahun 2008." Indonesia, 2008. [Online]. Available: https://www.dpr.go.id/

[11] P. Wanda, M. E. Hiswati, M. Diqi, and R. Herlinda, "Re-Fake: Klasifikasi Akun Palsu di Sosial Media Online menggunakan Algoritma RNN," *senastindo*, vol. 3, pp. 191–200, Dec. 2021, doi: 10.54706/senastindo.v3.2021.139.

[12] H. Kurniawan, "Deteksi Twitter Bot Menggunakan Klasifikasi Decision Tree," vol. 09, no. 01, 2020.

[13] H. Gunawan and G. S. Budhi, "Penerapan Machine Learning dalam mendeteksi Fake Account pada Instagram," 2022.

[14] Mustofa, A., Humaira, F. M., Ermawati, M., Natasari, P. S., Kurdianto, A. A., Prasetyo, A. A., & Faisal, A. L. F. (2023). TWITTER BUZZER DETECTION SYSTEM USING TWEET SIMILARITY FEATURE AND SUPPORT VECTOR MACHINE. *NJCA (Nusantara Journal of Computers and Its Applications)*, *8*(1), 7-12.

[15] A. R. N. Aouichaoui, R. Al, J. Abildskov, and G. Sin, "Comparison of Group-Contribution and Machine Learning-based Property Prediction Models with Uncertainty Quantification," in *Computer Aided Chemical Engineering*, Elsevier, 2021, pp. 755–760. doi: 10.1016/B978-0-323-88506-5.50118-2.

[16] X.-Y. Liu, Y. Fang, L. Yang, Z. Li, and A. Walid, "High-performance tensor decompositions for compressing and accelerating deep neural networks," in *Tensors for Data Processing*, Elsevier, 2022, pp. 293–340. doi: 10.1016/B978-0-12-824447-0.00015-7.

[17] F. Nur Fajri, A. Tholib, and W. Yuliana, "Application of Machine Learning Algorithm for Determining Elective Courses in Informatics Study Program," *JuTISI*, vol. 8, no. 3, Dec. 2022, doi: 10.28932/jutisi.v8i3.3990.

[18] S. Syafudin, R. A. Nugraha, and K. Handayani, "Prediksi Status Pinjaman Bank dengan Deep Learning Neural Network (DNN)," vol. 7, 2021.

[19] J. Lim *et al.*, "Development of Dye Exhaustion Behavior Prediction Model using Deep Neural Network," in *Computer Aided Chemical Engineering*, Elsevier, 2022, pp. 1825–1830. doi: 10.1016/B978-0-323-85159-6.50304-3.

[20] W. S. Lestari and A. Halim, "Prediksi Kesuksesan Startup Menggunakan Deep Neural Network," *J. Sifo Mikrosk.*, vol. 23, no. 2, pp. 99–110, Oct. 2022, doi: 10.55601/jsm.v23i2.885.

[21] M. S. Wibawa, "Pengaruh Fungsi Aktivasi, Optimisasi dan Jumlah Epoch Terhadap Performa Jaringan Saraf Tiruan," 2017, doi: 10.13140/RG.2.2.21139.94241.

[22] W. Setiawan, *Deep Learning Menggunakan Convolutional Neural Network: Teori dan Aplikasi*. Malang: Media Nusa Creative, 2020.

[23] U. I. Lestari, A. Yusrotun Nadhiroh, and C. Novia, "PENERAPAN METODE K-NEAREST NEIGHBOR UNTUK SISTEM PENDUKUNG KEPUTUSAN IDENTIFIKASI PENYAKIT DIABETES MELITUS," *JATISI*, vol. 8, no. 4, pp. 2071–2082, Dec. 2021, doi: 10.35957/jatisi.v8i4.1235.

[24] D. Muchlizin Wahidillah, A. Tholib, and M. Muafi, "PENERAPAN ALGORITMA K-NEAREST NEIGHBOR UNTUK KLASIFIKASI RUMAH LAYAK ATAU TIDAK LAYAK HUNI (STUDI KASUS: DESA BULU KECAMATAN KRAKSAAN KABUPATEN PROBOLINGGO)," 2023, doi: https://doi.org/10.35316/jimi.v7i2.75-84.

[25] C. Haryanto, N. Rahaningsih, and F. Muhammad Basysyar, "KOMPARASI ALGORITMA MACHINE LEARNING DALAM MEMPREDIKSI HARGA RUMAH," *jati*, vol. 7, no. 1, pp. 533–539, Mar. 2023, doi: 10.36040/jati.v7i1.6343.