LAPORAN TUGAS BESAR 2 IF3170
INTELIGENSI ARTIFISIAL

Implementasi Algoritma Pembelajaran Mesin

Dosen Pengampu : Dr. Nur Ulfa Maulidevi, S.T, M.Sc.

Disusun oleh:

Konstan Aftop Anewata Ndruru	**(12822058)**
Imam Hanif Mulyarahman	**(13522030)**
Bryan Cornelius Lauwrence	**(13522033)**
Amalia Putri	**(13522042)**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

1. Pendahuluan

Pembelajaran mesin merupakan salah satu cabang dari kecerdasan buatan yang memungkinkan sistem untuk belajar dari data dan membuat prediksi atau keputusan tanpa diprogram secara eksplisit.

Dataset **UNSW-NB15** adalah kumpulan data lalu lintas jaringan yang mencakup berbagai jenis serangan siber dan aktivitas normal. Pada tugas ini, Anda diminta untuk mengimplementasikan algoritma pembelajaran mesin yang telah kalian pelajari di kuliah, yaitu **KNN, Gaussian Naive-Bayes, dan ID3** pada dataset **UNSW-NB15**. Rincian spesifikasi untuk tugas besar 2 dapat dilihat sebagai berikut:

1) Implementasi KNN *from scratch*.
    a) Minimal bisa menerima 2 input parameter
        i) Jumlah tetangga
        ii) Metrik jarak antar data point. Minimal dapat menerima 3 pilihan, yaitu Euclidean, Manhattan, dan Minkowski
2) Implementasi Gaussian Naive-Bayes *from scratch*.
3) Implementasi ID3 *from scratch*, termasuk pemrosesan data numerik sesuai materi yang dijelaskan dalam PPT kuliah.
4) Implementasi algoritma poin 1-3 menggunakan *scikit-learn*. Bandingkan hasil dari algoritma *from scratch* dan algoritma *scikit-learn*.
5) Model harus bisa di-save dan di-load. Implementasinya dibebaskan (misal menggunakan .txt, .pkl, dll).
6) [Bonus] Kaggle Submission.

Implementasi KNN, Gaussian Naive-Bayes, dan ID3 yang *from scratch* bisa dalam bentuk kelas-kelas (class KNN, dst.) yang nantinya akan di-import ke notebook pengerjaan.

2. Implementasi KNN

Algoritma K-Nearest Neighbor (KNN) adalah algoritma machine learning yang bersifat non-parametric dan lazy learning. Metode yang bersifat non-parametric memiliki makna bahwa metode tersebut tidak membuat asumsi apa pun tentang distribusi data yang mendasarinya. Algoritma non-parametric menggunakan sejumlah parameter yang fleksibel, dan jumlah parameter seringkali bertambah seiring data yang semakin banyak. Algoritma non-parametric secara komputasi lebih lambat, tetapi membuat lebih sedikit asumsi tentang data. Algoritma KNN juga bersifat lazy learning, yang artinya tidak menggunakan titik data training untuk membuat model. KNN menggunakan semua data yang tersedia

dan mengklasifikasikan data atau kasus baru berdasarkan ukuran kesamaan atau fungsi jarak. Data baru kemudian ditugaskan ke kelas tempat sebagian besar data tetangga berada.

Algoritma ini menggunakan 3 fungsi jarak yang utama yaitu :

1) Euclidean Distance

Euclidean Distance adalah metrik yang mengukur jarak lurus (linear) terpendek antara dua titik dalam ruang Euclidean. Jarak ini dihitung menggunakan akar kuadrat dari jumlah selisih kuadrat setiap koordinat antara dua titik. Euclidean distance sering digunakan karena cocok untuk data yang memiliki hubungan geometris, tetapi sensitif terhadap perbedaan skala data, sehingga normalisasi sering diperlukan. Formulanya adalah sebagai berikut :

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

2) Manhattan Distance

Manhattan Distance mengukur jarak antara dua titik berdasarkan jalur horizontal dan vertikal saja, mirip dengan pola pergerakan di jalanan grid seperti di Manhattan. Metode ini lebih robust terhadap perubahan kecil dalam data dibandingkan Euclidean dan cocok untuk data diskrit atau berbentuk grid. Namun, metrik ini kurang ideal jika hubungan diagonal lebih relevan. Formulanya adalah sebagai berikut :

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

3) Minkowski Distance

Minkowski Distance adalah generalisasi dari Euclidean dan Manhattan distance, dengan parameter ppp yang dapat disesuaikan. Metrik ini sangat fleksibel, tetapi pemilihan nilai ppp harus mempertimbangkan konteks data

karena mempengaruhi sensitivitas terhadap pola data. Formulanya adalah sebagai berikut :

$$d = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

Pada pengimplementasian kode KNN di tugas besar kali ini, data yang digunakan dalam pengimplementasiannya hanyalah 100 baris dari data yang sudah dibersihkan. Hal ini disebabkan oleh proses pemodelan yang lambat sehingga kami memutuskan untuk menggunakan hanya sedikit dataset yang dibandingkan.

Fungsi/Prosedur	Penjelasan
__init__(self)	Konstruktor kelas yang dijalankan saat objek KNN dibuat.
fit(self, X, y):	Melatih model dengan menyimpan parameter matriks distance dan melakukan label encoding pada target.
predict(self, X):	Mengembalikan nilai prediksi dari setiap data uji.
_euclidean_distance(self, x1, x2)	Menghitung jarak menggunakan rumus euclidean distance.
_manhattan_distance(self, x1,x2)	Menghitung jarak menggunakan rumus manhattan distance.
_minkowski_distance(self, x1, x2)	Menghitung jarak menggunakan rumus minkowski distance.
save(self, filename)	Menyimpan model ke file
load(filename)	Memuat model dalam file

3. Implementasi Gaussian Naive-Bayes

Naive Bayes adalah keluarga algoritma pembelajaran mesin probabilistik yang didasarkan pada Teorema Bayes dengan asumsi independensi di antara fitur-fiturnya. Pengklasifikasi Naive Bayes mengasumsikan bahwa keberadaan

fitur dalam suatu kelas tidak terkait dengan fitur lainnya. Naive Bayes adalah algoritma klasifikasi untuk masalah klasifikasi biner dan multikelas.

Fungsi/Prosedur	Penjelasan
__init__(self)	Konstruktor kelas yang dijalankan saat objek NaiveBayes dibuat.
fit(self, X_train, y_train)	Melatih model dengan menghitung parameter distribusi Gaussian (mean dan variansi) serta probabilitas prior untuk setiap kelas.
predict(self, X_test)	Membuat prediksi untuk data uji. Menghitung posterior untuk setiap kelas berdasarkan data masukan (X_test) dan memilih kelas dengan nilai posterior tertinggi sebagai prediksi.
_predict(self, x)	Membuat prediksi untuk satu sampel data (x).
_gaussian_pdf(self, x, c)	Menghitung probabilitas densitas Gaussian (P(x|C)) untuk setiap fitur dalam sampel x, dengan asumsi fitur mengikuti distribusi Gaussian.
save_model(self, filename)	Menyimpan probabilitas yang telah dibuat pada filename dengan ekstensi .pkt
load_model(self, filename)	Memuat model probabilitas dari filename dengan ekstensi .pkt

4. Implementasi ID3

ID3 adalah salah satu algoritma pembelajaran yang bertujuan membuat *decision tree learning*. ID3 memanfaatkan pencarian dari atas ke bawah menggunakan metode *greedy*. Awalnya dipilih sebuah fitur yang memisahkan kelas dengan cara lebih baik. Proses tersebut diteruskan sampai diperoleh *leaf node*. Cara menentukan atribut terbaik menggunakan *entropy* dan *information gain*. Semakin besar *entropy* semakin besar juga keberagaman datanya. *Information gain* yang lebih besar artinya akan menurunkan nilai entropy. Algoritma ID3 secara rekursif akan menentukan *information gain* dari seluruh

fitur yang tersedia. Fitur dengan *information gain* tertinggi akan dijadikan *node* saat itu dengan *root*-nya masing-masing adalah nilai unik dari fitur tersebut. Klasifikasi *node* diteruskan secara rekursif. Kondisi basis dari algoritma ID3 adalah apabila semua kelas sama maka akan mengembalikan nilai kelas tersebut, apabila atribut kosong makan akan mengembalikan kelas terbanyak dari *node*, dan apabila sudah tidak ada data maka akan mengembalikan nilai terbanyak dari *parent node*. Salah satu masalah yang dihadapi ID3 adalah data numerik. Solusinya adalah dengan membagi data menjadi dua bagian berdasarkan suatu nilai dari data numerik tersebut. Namun, implementasi ID3 hanya menggunakan maksimal 10000 data untuk pelatihan model karena memerlukan waktu yang lama untuk proses pelatihan. Hal ini disebabkan karena perhitungan ID3 memerlukan banyak proses, terutama pada pencarian titik untuk diskritisasi yang perlu melakukan pengurutan data serta pencarian titik potong terbaik.

Fungsi/Prosedur	Penjelasan
__init__(self, X_train, y_train)	Konstruktor kelas yang dijalankan saat objek ID3 dibuat. Menerima X dan y yang akan menjadi model latihan
fit(self)	Melatih model dengan mengimplementasikan algoritma ID3 dan menyimpan *decision tree*
predict(self, X)	Membuat prediksi untuk data uji X. Setiap baris dari X akan ditentukan kelasnya berdasarkan *decision tree* yang tersimpan
_predict_row(self, tree, row)	Membuat prediksi untuk satu sampel data (x).
entropy(self, data, target_column)	Menghitung nilai *entropy* dengan input berupa dataset dan kolom yang menjadi kelas
_plurality_value(self, data, target_column)	Mengembalikan kelas terbanyak dari data
_find_best_split(self, data, feature, target_column)	Mencari batas data numerik terbaik untuk menentukan pembagian *root*
_importance(self, data, feature,	Menghitung *information gain* dengan

target_column)	mengeliminasi nilai dari fitur tertentu
_decision_tree_learning(self, data, attributes, target_column, parent_data=None)	Membuat pohon keputusan secara rekursif
save_model(self, filename)	Menyimpan model *decision tree* dalam filename yang berkesktensi .pkt
load_model(self, filename)	Memuat model *decision tree* dari filename dengan ekstensi .pkt

5. Persiapan Data

Pada tahap ini, data akan dipersiapkan agar dapat dilatih dengan baik oleh algoritma. Pertama, dilakukan EDA untuk mengetahui dan memahami data. Dari proses EDA, ditemukan bahwa data memiliki kelas yang tidak seimbang antara 10 kelas. Kemudian, diperiksa juga nilai unik beberapa fitur. Terdapat beberapa fitur kategorik yang memiliki kategori berjumlah satu yang mengindikasikan ada kemungkinan masalah saat melakukan *one hot encoding* saat dilakukan split pada data train dan data validation. Dalam tahap eksplorasi data, diperiksa juga missing values, outliers, dan duplikasi pada data yang berpotensi memperburuk performa model. Di bawah ini dilampirkan langkah lanjutan yang dilakukan dalam penanganan terhadap masalah yang ditemukan pada proses EDA.

Masalah / Persiapan	Perlakuan	Implementasi
Missing data	A. Imputasi Modus pada fitur kategorik. B. Imputasi median pada fitur numerik.	Class FeatureImputer()
Outliers	Awalnya, menggunakan teknik capping. Namun, performa model menurun. Sehingga, penanganannya saat scaling.	Class CappingOutliers()
Duplicated values	Menghapus datum dengan nilai duplikat.	Class RemoveDuplicates()

Feature Selection	Dihitung korelasi antar fitur numerik, fitur fitur yang memiliki korelasi tinggi, akan dipilih satu untuk mewakili. Hal ini untuk mencegah *multicolinearity*.	Class SelectFeatures()
Feature Scaling	Dilakukan scaling untuk fitur numerik dengan RobustScaler() yang cocok untuk data dengan outliers.	Class FeatureScaling()
Feature Encoding	Dilakukan encoding untuk fitur kategorik dengan OneHotEncoder dikarenakan fitur kategorik yang ada bersifat nominal.	Class FeatureEncoder()
Imbalanced Data	Dilakukan random over sampler untuk menghasilkan sample baru dari data dengan kelas minoritas.	Def handleimbalance()
Dimensionality Reduction	Dengan teknik PCA, untuk mengambil komponen yang esensial dari fitur yang banyak hasil encoding.	PCA(n_components=5)

6. Perbandingan Model Sklearn VS Model From Scratch

Model	Sklearn	Scratch
KNN Minkowski	Classification Report: precision recall f1-score support Analysis 0.00 0.00 0.00 380 Backdoor 0.00 0.00 0.00 365 DoS 0.22 0.15 0.18 2233 Exploits 0.29 0.41 0.34 6277 Fuzzers 0.23 0.36 0.28 3618 Generic 0.82 0.87 0.84 6636 Normal 0.60 0.54 0.57 11164 Reconnaissance 0.00 0.00 0.00 1894 Shellcode 0.00 0.00 0.00 259 Worms 0.00 0.00 0.00 28 accuracy 0.49 32854 macro avg 0.22 0.23 0.22 32854 weighted avg 0.47 0.49 0.47 32854	Classification Report: precision recall f1-score support cell output actions Analysis 0.00 0.00 0.00 380 Backdoor 0.00 0.00 0.00 365 DoS 0.22 0.15 0.18 2233 Exploits 0.29 0.41 0.34 6277 Fuzzers 0.23 0.36 0.28 3618 Generic 0.82 0.87 0.84 6636 Normal 0.60 0.54 0.57 11164 Reconnaissance 0.00 0.00 0.00 1894 Shellcode 0.00 0.00 0.00 259 Worms 0.00 0.00 0.00 28 accuracy 0.49 32854 macro avg 0.22 0.23 0.22 32854 weighted avg 0.47 0.49 0.47 32854

<table>
<tr><td>KNN Manhattan</td><td>

Classification Report:

	precision	recall	f1-score	support
Analysis	0.00	0.00	0.00	380
Backdoor	0.00	0.00	0.00	365
DoS	0.21	0.15	0.17	2233
Exploits	0.30	0.38	0.33	6277
Fuzzers	0.21	0.35	0.26	3618
Generic	0.82	0.87	0.85	6636
Normal	0.65	0.59	0.62	11164
Reconnaissance	0.00	0.00	0.00	1894
Shellcode	0.00	0.00	0.00	259
Worms	0.00	0.00	0.00	28
accuracy			0.50	32854
macro avg	0.22	0.23	0.22	32854
weighted avg	0.48	0.50	0.49	32854

</td><td>

Classification Report:

	precision	recall	f1-score	support
Analysis	0.00	0.00	0.00	380
Backdoor	0.00	0.00	0.00	365
DoS	0.21	0.15	0.17	2233
Exploits	0.30	0.38	0.33	6277
Fuzzers	0.21	0.35	0.26	3618
Generic	0.82	0.87	0.85	6636
Normal	0.65	0.59	0.62	11164
Reconnaissance	0.00	0.00	0.00	1894
Shellcode	0.00	0.00	0.00	259
Worms	0.00	0.00	0.00	28
accuracy			0.50	32854
macro avg	0.22	0.23	0.22	32854
weighted avg	0.48	0.50	0.49	32854

</td></tr>
<tr><td>KNN Euclidean</td><td>

Classification Report:

	precision	recall	f1-score	support
Analysis	0.00	0.00	0.00	380
Backdoor	0.00	0.00	0.00	365
DoS	0.22	0.15	0.17	2233
Exploits	0.33	0.39	0.36	6277
Fuzzers	0.23	0.35	0.28	3618
Generic	0.82	0.87	0.84	6636
Normal	0.63	0.64	0.63	11164
Reconnaissance	0.00	0.00	0.00	1894
Shellcode	0.00	0.00	0.00	259
Worms	0.00	0.00	0.00	28
accuracy			0.52	32854
macro avg	0.22	0.24	0.23	32854
weighted avg	0.48	0.52	0.50	32854

</td><td>

	precision	recall	f1-score	support
Analysis	0.00	0.00	0.00	380
Backdoor	0.00	0.00	0.00	365
DoS	0.22	0.15	0.17	2233
Exploits	0.33	0.39	0.36	6277
Fuzzers	0.23	0.35	0.28	3618
Generic	0.82	0.87	0.84	6636
Normal	0.63	0.64	0.63	11164
Reconnaissance	0.00	0.00	0.00	1894
Shellcode	0.00	0.00	0.00	259
Worms	0.00	0.00	0.00	28
accuracy			0.52	32854
macro avg	0.22	0.24	0.23	32854
weighted avg	0.48	0.52	0.50	32854

</td></tr>
<tr><td>Insight KNN</td><td colspan="2">Insight yang diperoleh dari perbandingan KNN menggunakan library sklearn dengan fungsi buatan sendiri adalah proses KNN from scratch membutuhkan banyak penyesuaian, terutama dalam jumlah datanya karena prosesnya yang lama. Untuk hasilnya sendiri mempunyai tingkat akurasi yang hampir sama dengan parameter k yang sama dengan KNN dengan library. Hal ini mungkin disebabkan karena algoritma KNN sangat sederhana dan mudah diimplementasikan versi manual. Perbedaan yang mencolok antara versi scratch dan library terletak pada kecepatan prosesnya.</td></tr>
<tr><td>Naive Bayes</td><td>

Classification Report:

	precision	recall	f1-score	support
Analysis	0.02	0.28	0.04	380
Backdoor	0.02	0.01	0.01	365
DoS	0.04	0.00	0.01	2233
Exploits	0.59	0.02	0.03	6277
Fuzzers	0.07	0.06	0.06	3618
Generic	0.92	0.32	0.47	6636
Normal	1.00	0.05	0.09	11164
Reconnaissance	0.00	0.00	0.00	1894
Shellcode	0.01	1.00	0.03	259
Worms	0.01	0.14	0.03	28
accuracy			0.10	32854
macro avg	0.27	0.19	0.08	32854
weighted avg	0.65	0.10	0.14	32854

</td><td>

Classification Report:

	precision	recall	f1-score	support
Analysis	0.02	0.34	0.05	380
Backdoor	0.02	0.02	0.02	365
DoS	0.03	0.00	0.00	2233
Exploits	0.58	0.02	0.04	6277
Fuzzers	0.08	0.07	0.08	3618
Generic	0.93	0.68	0.78	6636
Normal	0.98	0.10	0.17	11164
Reconnaissance	0.00	0.00	0.00	1894
Shellcode	0.02	1.00	0.03	259
Worms	0.00	0.14	0.01	28
accuracy			0.19	32854
macro avg	0.27	0.24	0.12	32854
weighted avg	0.65	0.19	0.23	32854

</td></tr>
<tr><td>Insight Naive Bayes</td><td colspan="2">Implementasi Naive Bayes dari scratch menunjukkan performa lebih baik dalam accuracy dan macro average F1-score dibandingkan Scikit-learn. Namun, perlu analisis lebih mendalam untuk memastikan bahwa semua aspek model (seperti smoothing dan normalisasi) diimplementasikan dengan benar pada versi manual.</td></tr>
</table>

<table>
<tr><td>ID3</td><td>Classification Report:
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Analysis</td><td>0.11</td><td>0.08</td><td>0.09</td><td>380</td></tr>
<tr><td>Backdoor</td><td>0.09</td><td>0.05</td><td>0.07</td><td>365</td></tr>
<tr><td>DoS</td><td>0.28</td><td>0.26</td><td>0.27</td><td>2233</td></tr>
<tr><td>Exploits</td><td>0.59</td><td>0.66</td><td>0.62</td><td>6277</td></tr>
<tr><td>Fuzzers</td><td>0.57</td><td>0.57</td><td>0.57</td><td>3618</td></tr>
<tr><td>Generic</td><td>0.98</td><td>0.97</td><td>0.98</td><td>6636</td></tr>
<tr><td>Normal</td><td>0.86</td><td>0.86</td><td>0.86</td><td>11164</td></tr>
<tr><td>Reconnaissance</td><td>0.74</td><td>0.68</td><td>0.71</td><td>1894</td></tr>
<tr><td>Shellcode</td><td>0.35</td><td>0.36</td><td>0.36</td><td>259</td></tr>
<tr><td>Worms</td><td>0.21</td><td>0.14</td><td>0.17</td><td>28</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.74</td><td>32854</td></tr>
<tr><td>macro avg</td><td>0.48</td><td>0.46</td><td>0.47</td><td>32854</td></tr>
<tr><td>weighted avg</td><td>0.73</td><td>0.74</td><td>0.74</td><td>32854</td></tr>
</table></td><td>Classification Report:
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Analysis</td><td>0.00</td><td>0.00</td><td>0.00</td><td>380</td></tr>
<tr><td>Backdoor</td><td>0.00</td><td>0.00</td><td>0.00</td><td>365</td></tr>
<tr><td>DoS</td><td>0.33</td><td>0.13</td><td>0.19</td><td>2233</td></tr>
<tr><td>Exploits</td><td>0.44</td><td>0.77</td><td>0.56</td><td>6277</td></tr>
<tr><td>Fuzzers</td><td>0.60</td><td>0.09</td><td>0.16</td><td>3618</td></tr>
<tr><td>Generic</td><td>0.97</td><td>0.97</td><td>0.97</td><td>6636</td></tr>
<tr><td>Normal</td><td>0.72</td><td>0.81</td><td>0.76</td><td>11164</td></tr>
<tr><td>Reconnaissance</td><td>0.50</td><td>0.33</td><td>0.40</td><td>1894</td></tr>
<tr><td>Shellcode</td><td>1.00</td><td>0.01</td><td>0.02</td><td>259</td></tr>
<tr><td>Worms</td><td>0.00</td><td>0.00</td><td>0.00</td><td>28</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.65</td><td>32854</td></tr>
<tr><td>macro avg</td><td>0.46</td><td>0.31</td><td>0.30</td><td>32854</td></tr>
<tr><td>weighted avg</td><td>0.65</td><td>0.65</td><td>0.61</td><td>32854</td></tr>
</table></td></tr>
<tr><td>Insight ID3</td><td colspan="2">Insight yang diperoleh dari perbandingan ID3 menggunakan library sklearn dengan fungsi buatan sendiri adalah proses ID3 memerlukan banyak penyesuaian dan optimasi karena prosesnya akan bertambah dengan bertambahnya jumlah nilai unik dan perlu mempertimbangkan banyak hal lain supaya model yang dihasilkan tidak overfit.</td></tr>
</table>

DAFTAR PUSTAKA

1. Stackoverflow. “How to compute correlation ratio or Eta in Python?” (online).
   (https://stackoverflow.com/questions/52083501/how-to-compute-correlation-ratio-or-eta-in-python, diakses pada 1 Desember 2024).
2. Brownlee, Jason. “How to Choose a Feature Selection Method For Machine Learning” (online).
   (https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/, diakses pada 1 Desember 2024).
3. Geeksforgeeks. “ML | Naive Bayes Scratch Implementation using Python” (online).
   (https://www.geeksforgeeks.org/ml-naive-bayes-scratch-implementation-using-python/, diakses pada 1 Desember 2024).
4. Geeksforgeeks. “Sklearn | Iterative Dichotomiser 3 (ID3) Algorithms” (online).
   (https://www.geeksforgeeks.org/sklearn-iterative-dichotomiser-3-id3-algorithms/, diakses pada 13 Desember 2024).
5. Russel, S. and Norving, P. 2003. “Artificial Intelligence A Modern Approach Third Edition”. Pearson: New Jersey.

LAMPIRAN

Pengecekan Program

Poin	Ya	Tidak
1. Implementasi KNN	✓	
2. Implementasi Naive Bayes	✓	
3. Implementasi ID3	✓	
4. Implementasi menggunakan *scikit-learn*	✓	
5. Model dapat di-*save* dan *load*	✓	
7. [Bonus]: Submission *kaggle*	✓	

Tabel 3. Tabel Pengecekan Program

Lembar Kontribusi

Nama (NIM)	Kontribusi
Konstan Aftop Anewata Ndruru (12822058)	Data Preparation + Implementasi Scikit-learn
Imam Hanif Mulyarahman (13522030)	KNN from scratch
Bryan Cornelius Lauwrence (13522033)	ID3 from scratch
Amalia Putri (13522042)	Naive Bayes from scratch

Tabel 4. Lembar Kontribusi

Repository

Link Repository dari Tugas Besar 2 Inteligensi artifisial adalah sebagai berikut.

https://github.com/amaliap21/Data-Enjoyer.git