Tugas Besar 2

IF3170 Intelegensi Artifisial

Implementasi Algoritma Pembelajaran Mesin

Disusun oleh :

Ahmad Farid Mudrika	13522008/K01
Muhammad Yusuf Rafi	13522009/K01
Naufal Baldemar	13521154/?

PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG

2024

I. Deskripsi Persoalan

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*. Untuk ID3 di *scikit-learn*, gunakan DecisionTreeClassifier dengan parameter criterion='entropy' (memang tidak sama persis dengan ID3, tetapi cukup mendekati)
5. Model harus bisa di-save dan di-load. Implementasinya dibebaskan (misal menggunakan .txt, .pkl, dll).
6. [Bonus] Kaggle Submission pada link berikut.

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. Untuk implementasi *from scratch*, *library* yang boleh digunakan adalah untuk perhitungan matematika saja seperti numpy dan sejenisnya.

II. Pembahasan

1. K Nearest Neighbor

   Algoritma K-Nearest Neighbor (KNN) adalah metode klasifikasi sederhana dalam pembelajaran mesin. Prinsip kerjanya didasarkan pada premis bahwa data yang berdekatan cenderung memiliki kelas yang sama.

   Proses kerja KNN secara umum adalah sebagai berikut:

   1. Perhitungan Jarak

      Pilih data baru yang ingin diklasifikasi. Hitung jarak dengan euclidean distance. Agar tidak ada fitur yang menghasilkan jarak terlalu besar, lakukan scaling pada data baru. Scaling yang kami gunakan adalah standardization scaler
   2. Penentuan Tetangga Terdekat

      Urutkan data pelatihan berdasarkan jaraknya terhadap data baru dari terdekat hingga terjauh. Lalu, ambil k data pelatihan terdekat sebagai tetangga terdekat. Nilai k merupakan parameter yang perlu ditentukan dan berpengaruh pada hasil prediksi.
   3. Klasifikasi

      Tentukan kelas dari data baru berdasarkan kelas mayoritas dari k tetangga terdekat. Jika semua tetangga memiliki kelas yang sama, maka data baru akan diklasifikasikan ke kelas tersebut. Jika tidak, maka akan dilakukan voting untuk memilih kelas yang paling banyak muncul.

2. Naive Bayes

   Algoritma Naive Bayes adalah algoritma klasifikasi yang didasarkan pada Teorema Bayes dengan asumsi bahwa setiap fitur dalam dataset bersifat independen satu sama lain. Keunggulan pengkasifikasi Naive Bayes adalah kemampuan prediksinya yang cepat. Salah satu jenis model Naive Bayes yang kami gunakan untuk program ini adalah Gaussian Naive Bayes. Dalam Gaussian Naive Bayes, nilai kontinu yang dikaitkan dengan setiap fitur diasumsikan terdistribusi normal.

   Proses kerja Gaussian Naive Bayes adalah sebagai berikut:

   1. Perhitungan Mean, Variance, dan Prior

      Menghitung mean dan variance untuk setiap fitur dalam setiap kelas. Lalu, menghitung prior untuk setiap kelas,
   2. Perhitungan probabilitas fitur kontinu (likelihood)

      Untuk tiap fitur, hitung probabilitas fitur berdasarkan nilai mean dan variansi yang telah dihitung sebelumnya.
   3. Perhitungan Probabilitas Posterior

      Setelah menghitung prior dan likelihood, hitung probabilitas posterior setiap kelas berdasarkan nilai mean dan variansi yang telah dihitung sebelumnya. Juga, perhitungan dilakukan dalam bentuk log untuk menghindari *underflow.*
   4. Penentuan Prediksi Kelas

      Kelas dengan probabilitas posterior tertinggi akan dipilih

3. ID3

   Algoritma ID3 (Iterative Dichotomiser 3) adalah salah satu metode pembelajaran mesin yang digunakan untuk membangun pohon keputusan. Pohon keputusan ini digunakan untuk melakukan klasifikasi atau prediksi terhadap data baru.

   Cara kerja ID3 adalah sebagai berikut:

   1. Memilih atribut terbaik

      ID3 menggunakan konsep entropy untuk mengukur ketidakpastian atau kemurnian dari suatu kumpulan data. Semakin tinggi entropy, semakin sulit untuk membuat keputusan yang akurat. ID3 kemudian menghitung information gain untuk setiap atribut. Information gain menunjukkan seberapa banyak ketidakpastian(entropy) berkurang jika kita menggunakan atribut tersebut untuk membuat keputusan. Atribut dengan information gain tertinggi akan dipilih sebagai akar pohon.
   2. Membuat cabang

      Setelah memilih akar, data dibagi menjadi beberapa cabang berdasarkan nilai atribut pada akar. Proses memilih atribut terbaik dan membuat cabang diulang secara rekursif untuk setiap cabang baru hingga semua data terklasifikasi atau mencapai kriteria penghentian lain(contohnya, kedalaman pohon maksimum).
   3. Membentuk pohon

      Proses berulang ini akan menghasilkan sebuah pohon keputusan. Setiap cabang dalam pohon mewakili suatu keputusan, dan daun dari pohon mewakili kelas atau prediksi akhir.

4. Cleaning and Preprocessor

   Untuk Data Cleaning, kami lakukan 4 proses, diantaranya

   1. Handling Missing Data

      Kami menggunakan Imputation Libraries, yanki SimpleImputer dengan strategi ‘mean’ dari Scikit-Learn untuk numerikal data. Kami memlih strategi ‘mean’ karena memberikan estimasi yang cukup representatid dan juga menjaga kestabilan distribusi data . Adapun untuk kategorikal data, kami menggunakan strategi ‘most_frequent’ karena mencegah perubahan distribusi data dan cukup sesuai untuk data diskrit.
   2. Dealing with Outliers

      Kami menggunakan metode Clipping karena metode ini membantu mencegah outlier ekstrem yang dapat mendistorsi model, di mana nilai-nilai di luar rentang yang telah ditentukan akan dipotong. Dengan multiplier sebesar 1,5, nilai yang lebih kecil dari batas bawah akan digantikan dengan batas bawah, dan nilai yang lebih besar dari batas atas akan digantikan dengan batas atas.
   3. Removing Duplicates

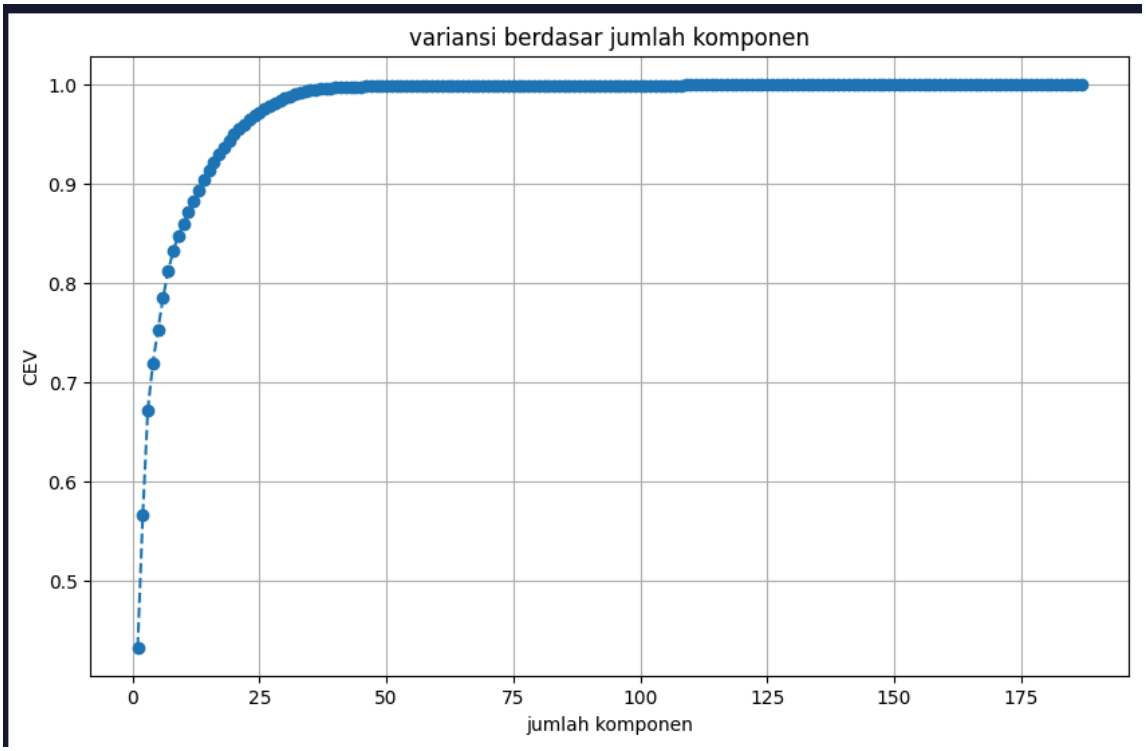
      Kami lakukan penghapusan data yang terduplikasi pada data.
   4. Feature Engineering

Kami lakukan feature selection, dengan drop feature pada 'synack' dan ‘'ackdat' karena sudah ada feature ‘tcprtt’ yang berisikan penjumlahan 'synack' dan ‘ackdat'. Lalu, kami juga melakukan create new feature, yakni feature ‘jit’ beirisikan penjumlahan feature 'sjit' dan 'djit'. Lalu, drop kedua feature tersebut setelah menjumlahkannya. Terakhir, kami drop feature ‘Label’ karena tidak relevan.

Untuk Data Preprocessing, kami lakukan 4 proses berikut:

1. Feature Scaling
   Kami menggunakan Standardization (Z-score Scaling) karena menjaga distribusi nilai data tetap stabil, dimana dengan setiap fitur dipusatkan pada nilai rata-rata dan dibagi dengan standar deviasi standar, hal ini memastikan setiap feature memiliki rata-rata 0 dan standar deviasi 1.
2. Encoding Categorical Variables
   Kami menggunakan One-Hot Encoding karena mengubah variabel kategorikal menjadi representasi numerik, di mana setiap kategori dalam variabel, dipetakan menjadi kolom biner (0/1). Terutama, data kami juga tidak memiliki relasi ordinal sehingga cukup sesuai dan tidak membingungkan model nantinya.
3. Handling Imbalanced Classes
   Kami menggunakan SMOTE (Synthetic Minority Over-sampling Technique), yakni oversampling karena tidak hanya menambah contoh data minoritas yang ada, tapi juga menciptakan contoh sintetik baru dengan menginterpolasi antara sampel minoritas yang ada sehingga mencegah overfitting pada data minoritas dan memberikan distribusi data yang lebih baik.
4. Dimensionality Reduction

Kami melakukan plotting variansi berdasar jumlah komponen. Berikut merupakan grafiknya



Dari grafik tersebut, kami menilai bahwa 22 buah komponen telah mewakili lebih dari 95% variansi. Jadi, kami memasukkan PCA dengan n_components=22 ke dalam pipeline kami.

5. Perbandingan Hasil Implementasi dan Hasil yang Didapatkan dari Pustaka
   Kami menggunakan fungsi accuracy_score dari sklearn.metrics untuk menghitung akurasi prediksi.
   a. KNN

Berikut merupakan akurasi yang didapat dari implementasi KNN yang kami buat.

Berikut merupakan akurasi knn dari pustaka.

```python
# Type your code here
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

knn = KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train, y_train)

# Prediksi dengan data validasi
y_pred_knn = knn.predict(X_val)
print("Akurasi KNN:", accuracy_score(y_val, y_pred_knn))
print("Classification Report KNN:\n", classification_report(y_val, y_pred_knn))
```

```
Akurasi KNN: 0.7471556075166101
```

Dapat dilihat bahwa akurasi implementasi cukup dekat, bahkan melebihi akurasi *library*. Akan tetapi, waktu yang dibutuhkan implementasi kami untuk melakukan prediksi adalah sekitar 15 menit, sedangkan model dari *library* hanya memerlukan beberapa detik.

b. Naive Bayes

Berikut merupakan akurasi dari implementasi Naive Bayes kami.

```
Accuracy: 0.6650032792494796
Classification Report:
               precision    recall  f1-score   support

      Analysis       0.00      0.00      0.00       384
      Backdoor       0.06      0.03      0.04       367
           DoS       0.26      0.68      0.38      2459
      Exploits       0.63      0.43      0.51      6636
       Fuzzers       0.41      0.40      0.41      3637
       Generic       0.98      0.94      0.96      7975
        Normal       0.91      0.78      0.84     11254
Reconnaissance       0.34      0.51      0.41      2070
     Shellcode       0.25      0.00      0.01       259
         Worms       0.31      0.18      0.23        28

      accuracy                           0.67     35069
     macro avg       0.42      0.40      0.38     35069
  weighted avg       0.72      0.67      0.68     35069
```

Berikut merupakan akurasi implementasi Naive Bayes menggunakan library sklearn

```
Akurasi Naive Bayes: 0.6650032792494796
Classification Report Naive Bayes:
               precision    recall  f1-score   support

      Analysis       0.00      0.00      0.00       384
      Backdoor       0.06      0.03      0.04       367
           DoS       0.26      0.68      0.38      2459
      Exploits       0.63      0.43      0.51      6636
       Fuzzers       0.41      0.40      0.41      3637
       Generic       0.98      0.94      0.96      7975
        Normal       0.91      0.78      0.84     11254
Reconnaissance       0.34      0.51      0.41      2070
     Shellcode       0.25      0.00      0.01       259
         Worms       0.31      0.18      0.23        28

      accuracy                           0.67     35069
     macro avg       0.42      0.40      0.38     35069
  weighted avg       0.72      0.67      0.68     35069
```

Dapat dilihat bahwa akurasi dan laporan klasifikasi kedua model tersebut sama persis.

c. ID3

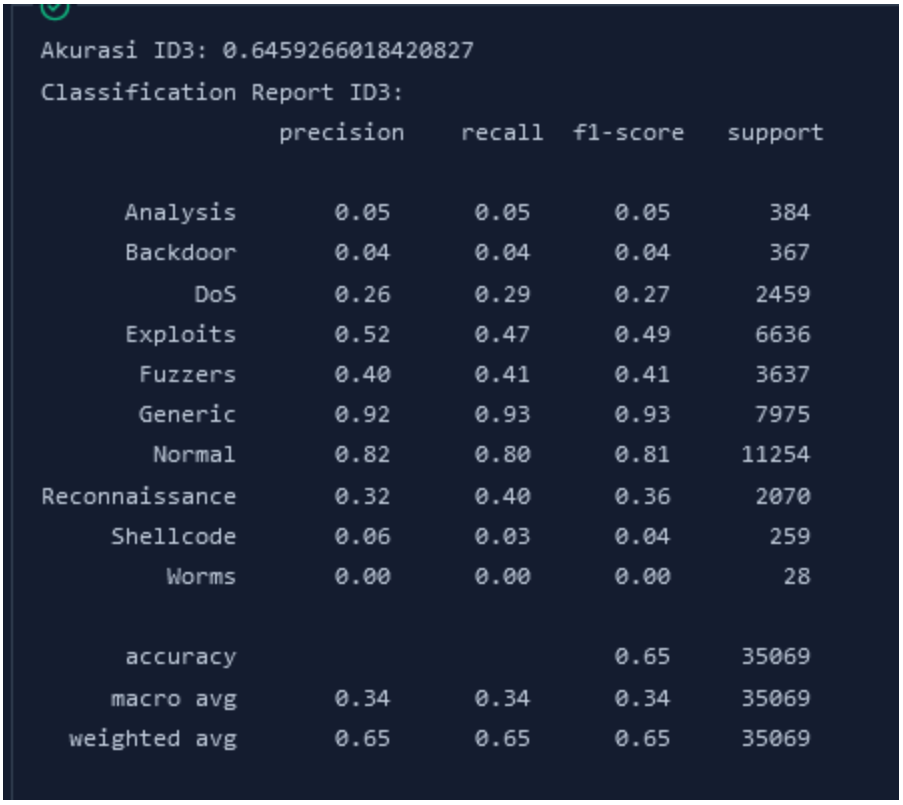
Dalam proses training, kami menggunakan 1000 baris data acak, karena waktu yang dibutuhkan untuk menggunakan keseluruhan data *training* terlalu lama. Model id3 dari library sebenarnya tidak membutuhkan pengurangan data, tetapi kami tetap melakukannya demi perbandingan.Berikut merupakan akurasi dari implementasi ID3 yang kami buat.

```
Accuracy: 0.6517437052667598
Classification Report:
/shared-libs/python3.11/py/lib/python3.11/site-packages/sk
  _warn_prf(average, modifier, msg_start, len(result))
                precision    recall  f1-score   support

      Analysis       0.02      0.01      0.01       384
      Backdoor       0.05      0.02      0.03       367
           DoS       0.03      0.00      0.00      2459
      Exploits       0.48      0.66      0.56      6636
       Fuzzers       0.41      0.45      0.43      3637
       Generic       0.95      0.92      0.93      7975
        Normal       0.94      0.75      0.83     11254
Reconnaissance       0.22      0.52      0.31      2070
     Shellcode       0.00      0.00      0.00       259
         Worms       0.00      0.00      0.00        28

      accuracy                           0.65     35069
     macro avg       0.31      0.33      0.31     35069
  weighted avg       0.67      0.65      0.65     35069
```

Berikut merupakan akurasi dari ID3 library menggunakan DecisionTreeClassifier dan entropy sebagai kriteria



Akurasi ID3: 0.6459266018420827

Classification Report ID3:

	precision	recall	f1-score	support
Analysis	0.05	0.05	0.05	384
Backdoor	0.04	0.04	0.04	367
DoS	0.26	0.29	0.27	2459
Exploits	0.52	0.47	0.49	6636
Fuzzers	0.40	0.41	0.41	3637
Generic	0.92	0.93	0.93	7975
Normal	0.82	0.80	0.81	11254
Reconnaissance	0.32	0.40	0.36	2070
Shellcode	0.06	0.03	0.04	259
Worms	0.00	0.00	0.00	28
accuracy			0.65	35069
macro avg	0.34	0.34	0.34	35069
weighted avg	0.65	0.65	0.65	35069

Dapat dilihat bahwa akurasi implementasi sedikit melebihi akurasi *library*.

III. Kesimpulan dan Saran

Dari semua algoritma yang kami terapkan, algoritma KNN membrikan akurasi yang paling besar dengan nilai 0,81 bahkan lebih besar dari penggunaan pustaka. Lalu, algoritma Gaussian Naive Bayes memberikan nilai akurasi terbesar ke-2 dengan nilai sekitar 0,66 yang sama persis dengan penggunaan pustaka. Terkahir, algoritma ID3 memberikan nilai akurasi terkecil sekitar 0.65 melebihi penggunaan pustaka. Selain pengembangan model dan pemilihan model terbaik, tahap data cleaning and preprocessing juga sangat penting, karena semakin sesuai data yang kami siapkan untuk pemrosesan machine learning, maka semakin optimal juga hasil model yang telah kami kembangkan.

IV. Pembagian Tugas

Ahmad Farid Mudrika	Cleaning, Preprocessor, knn, id3
Muhammad Yusuf Rafi	Cleaning, Preprocessor, nb
Naufar Baldemar	Hilang

V. Referensi

https://www.ibm.com/topics/knn
https://builtin.com/artificial-intelligence/gaussian-naive-bayes
https://builtin.com/artificial-intelligence/gaussian-naive-bayes