Laporan Tugas Besar 2 IF3170 Inteligensi Artifisial
Implementasi Algoritma Pembelajaran Mesin

Kelompok 4
Muhamad Rafli Rasyidin - 13522088
Julian Caleb Simandjuntak - 13522099
Christopher Brian - 13522106
Indraswara Galih Jayanegara - 13522119

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

1. **Implementasi KNN**

   Kelas KNN yang diimplementasikan sendiri memanfaatkan struktur data KDTree dari library scipy. Hal ini terpaksa dilakukan sebab membuat kelas KNN tanpa menggunakan struktur data dari library eksternal memakan waktu yang sangat lama. Dari pengalaman kami, waktu yang dibutuhkan untuk hanya 10 persen dari dataset membutuhkan waktu hingga 5 menit. Struktur data KDTree memungkinkan pengurutan tetangga terdekat secara efisien dan cepat. Perlu dicatat, kelas KNN implementasi sendiri menggunakan library sklearn hanya untuk memastikan kelas ini kompatibel dengan pipeline dan kode eksisting yang didesain awalnya untuk kelas KNN dari library sklearn.

   Kelas ini memiliki 4 metode, yaitu metode init, fit, predict, dan vote. Metode init akan menginisialisasi kelas KNN menggunakan struktur data KDTree. Metode ini menerima 2 parameter, yaitu k (jumlah tetangga terdekat) dan distance_metric (metrik jarak). Metrik jarak yang diperbolehkan yaitu euclidean, manhattan, dan chebyshev. Metode fit akan melakukan fitting terhadap model menggunakan training data dengan parameter berupa x (Fitur training) dan y (Label training). Metode predict akan memprediksi label kelas dari data input dengan parameter berupa X (Fitur testing) dan mengembalikan label kelas hasil prediksi. Metode vote akan mencari label kelas terbanyak dari tetangga terdekat dengan menerima parameter berupa labels (List atau array yang berisi label tetangga terdekat) dan mengembalikan label kelas hasil prediksi.

2. **Implementasi Naive-Bayes**

   Kelas Naive-Bayes yang kami implementasikan menggunakan distribusi gauss atau distribusi untuk mengatasi data numerik. Untuk data kategorikal, kami memprosesnya dengan Naive-Bayes seperti biasa.

   Pada kelas NaiveBayesClassifier, terdapat 3 fungsi utama, yaitu fit, gaussian, dan predict. Fungsi gaussian digunakan untuk menghitung probabilitas distribusi normal untuk setiap nilai pada kolom numerik. Ada cara lain untuk menangani data numerik menggunakan Naive-Bayes yaitu dengan mengelompokkan data numerik menjadi beberapa kategori (binning). Namun, kami memilih Gaussian karena implementasinya yang lebih mudah. Metode init pada kelas NaiveBayesClassifier berfungsi untuk menyimpan daftar kolom kategorikal dan numerik yang nantinya akan digunakan saat fit dan predict. Daftar kolom tersebut diperlukan agar Naive-Bayes tidak memproses data kategorikal seperti data numerik. Hal tersebut bisa saja terjadi karena data kategorikal berubah menjadi data numerik setelah melewati *encoder* sehingga kami memerlukan daftar kolom kategorikal dan numerik sebelum *encoding* dilakukan. Metode fit dan predict

masih sama seperti algoritma lainnya yang berfungsi untuk fitting model terhadap data latih dan mengembalikan hasil prediksi.

Kelas Naive-Bayes yang kami implementasikan lebih lambat dan kurang akurat dibandingkan dengan kelas GaussianNB pada scikit-learn. Perbedaan nilai akurasi tersebut mungkin disebabkan oleh perhitungan peluang yang saya lakukan. Pada saat mengecek probability, saya menemukan nilai None sehingga saya mengabaikan hal tersebut dan tetap melanjutkan perhitungan. Perbedaan ini bisa saja disebabkan oleh nilai None yang diabaikan tersebut.

```
# Type your code here
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
for i in range(len(X_train_array)):
    model.fit(X_train_array[i], y_train_array[i])

    y_pred = model.predict(X_val_array[i])
    accuracy = accuracy_score(y_pred, y_val_array[i])
    print(f"Fold {i+1}: Accuracy = {accuracy:.2f}")
```

[47] ✓ 3.0s

```
Fold 1: Accuracy = 0.43
Fold 2: Accuracy = 0.43
Fold 3: Accuracy = 0.43
Fold 4: Accuracy = 0.43
Fold 5: Accuracy = 0.43
```

```
categorical_col = df_combined.select_dtypes(include="object").columns.tolist()
categorical_col.remove('attack_cat')
numerical_col = df_combined.select_dtypes(include="number").columns.tolist()
numerical_col.remove('label')

nb = NaiveBayesClassifier(categorical_col, numerical_col)
for i in range(len(X_train_array)):
    nb.fit(X_train_array[i], y_train_array[i])

    y_pred = nb.predict(X_val_array[i])
    accuracy = accuracy_score(y_val_array[i], y_pred)
    print(f"Fold {i+1}: Accuracy = {accuracy:.2f}")
```

✓ 4m 47.3s

```
Fold 1: Accuracy = 0.43
Fold 2: Accuracy = 0.42
Fold 3: Accuracy = 0.43
Fold 4: Accuracy = 0.40
Fold 5: Accuracy = 0.40
```

3. **Implementasi ID3**

   Kelas ID3 yang kami implementasikan memanfaatkan struktur data Tree, di mana setiap Tree menggunakan kelas Node. Implementasi kelas ini relatif lebih lama dibandingkan dengan library bawaan Python. Selain itu, Tree yang kami buat menggunakan pendekatan rekursif, sebagaimana yang umum digunakan dalam algoritma pohon keputusan.

   Pada kelas ID3MultiClass yang kami implementasikan terdapat method calculate_multiclass_entropy untuk menghitung entropy, information_gain, most_common_label digunakan untuk mengisi None menjadi kelas yang paling umum, build_tree, fit digunakan untuk melatih, predict, predict_sample, dan evaluate untuk mengevaluasi hasil dari model. Sedangkan, untuk kelas Node berada di dalam kelas ID3MultiClass isinya sendiri hanya attribute seperti feature_name, decision, value, dan childs.

4. **Penjelasan tahap cleaning**
    a. **Handling missing data**

        Untuk handling missing data, kami menggunakan 2 fungsi, yaitu NumericalImputer dan CategoricalImputer. Fungsi NumericalImputer digunakan pada kolom bertipe numerikal untuk mengganti nilai yang kosong dengan rata-rata kolom. Sementara itu, fungsi CategoricalImputer digunakan pada kolom bertipe kategorikal untuk mengganti nilai yang kosong dengan nilai yang paling sering muncul pada kolom. Kedua fungsi ini memanfaatkan fungsi SimpleImputer dari library sklearn. Teknik ini kami pilih untuk memudahkan proses imputation serta tetap menjaga akurasi model untuk nilai kosong dengan tidak menggunakan teknik penghapusan missing data.
    b. **Dealing with outliers**

        Untuk outlier, kami bagi handling untuk data numerikal dan kategorikal. Untuk data numerikal, data yang bernilai di bawah Q1 - 1,5 * IQR digantikan dengan lower bound, sedangkan data yang bernilai di atas Q3 + 1,5 * IQR digantikan dengan upper bound. Untuk data kategorikal, khusus kolom state, nilai yang panjangnya tidak sama dengan 3 karakter dihapus dari data. Cara ini kami pilih karena seluruh nilai pada state memiliki pola 3 karakter sehingga nilai lain yang tidak mengikuti pola tersebut merupakan data yang *error*.
    c. **Remove duplicates**

        Remove duplicates dilakukan dengan melakukan drop pada row yang memiliki kesamaan dengan rows lainnya melalui fungsi yang dibuat berdasarkan fungsi bawaan pandas yaitu df.drop_duplicates. Remove duplicates dilakukan untuk menghindari bias pada model. Remove duplicates hanya dilakukan pada training set karena validation set hanya digunakan untuk mengecek sehingga tidak berpengaruh pada model.
    d. **Feature engineering**

        Terdapat beberapa fitur baru yang kami tambahkan untuk membantu akurasi model. Terdapat fitur baru yang berkaitan dengan rasio, yaitu byte_ratio, pkt_ratio, load_ratio, jit_ratio, inter_pkt_ratio, dan tcp_setup_ratio. Semuanya merupakan rasio antara nilai variabel source dan destination. Terdapat juga fitur baru yang berkaitan dengan agregat, yaitu total_bytes, total_pkts, total_load, total_jitter, total_inter_pkt, total_dan tcp_setup. Semuanya merupakan penjumlahan antara nilai variabel source dan destination. Terdapat juga fitur baru yang berkaitan dengan interaksi, yaitu byte_pkt_interaction_src, byte_pkt_interaction_dst, load_jit_interaction_src, load_jit_interaction_dst, pkt_jit_interaction_src, dan pkt_jit_interaction_dst. Semuanya merupakan hasil perkalian dari dua variabel. Terakhir, terdapat 1 fitur baru yang berkaitan dengan statistik, yaitu tcp_seq_diff yang berupa selisih antara variabel stcpb dan dtcpb. Pilihan fitur baru murni dari insting

dasar kami, bukan berdasarkan pengetahuan mendalam terhadap keilmuan yang terkait dengan dataset.

5. **Penjelasan tahap preprocessing**
   a. **Feature scaling**
      Untuk feature scaling, kami menggunakan fungsi MinMaxScaler dari library sklearn. Rentang yang kami gunakan adalah 0 sampai 1. Teknik ini kami pilih karena secara umum, teknik ini merupakan salah satu teknik yang paling banyak digunakan dan cenderung bekerja baik untuk kebanyakan dataset.
   b. **Feature encoding**
      Untuk feature encoder, kami menggunakan teknik label encoding untuk mengubah setiap kategori di dalam sebuah fitur kategorikal menjadi integer unik. Kami memanfaatkan fungsi LabelEncoder dari library sklearn. Teknik ini kami pilih karena one-hot encoding akan membutuhkan banyak sekali kolom mengingat jumlah nilai unik yang cukup banyak untuk fitur kategorikal, sehingga mahal secara komputasi.
   c. **Handling imbalanced data**
      Handling terhadap imbalanced data kami lakukan menggunakan teknik oversampling dengan memanfaatkan modul SMOTE dari library imblearn. Teknik ini kami lakukan karena teknik ini cukup banyak digunakan pada umumnya.
   d. **Dimensionality reduction**
   e. **Normalization**
      Teknik ini tidak kami lakukan secara khusus karena pada feature scaling, kami memilih teknik MinMaxScaler, yang pada dasarnya merupakan normalisasi.

6. **Pembagian tugas**

Anggota Kelompok	Kontribusi
Muhamad Rafli Rasyidin (13522088)	Handle Outlier, Feature Scaling, Naive Bayes
Julian Caleb Simandjuntak (13522099)	EDA, K-Fold Split, Remove Duplicate, Feature Encoding, Library (Scikit-Learn) Model, Submission
Christopher Brian (13522106)	EDA, Train-Test Split (not used), Feature Engineering, Data Normalization, KNN

Indraswara Galih Jayanegara (13522119)	Handle Missing Data, Handle Imbalance Data, ID3