TUGAS BESAR 2 INTELEGENSI ARTIFISIAL

IF3170

Implementasi Algoritma Pembelajaran Mesin

Semester V Tahun 2024/2025

Disusun oleh kelompok claude sonnet 3.6:

Ahmad Naufal Ramadan	: 13522005
Kristo Anugrah	: 13522024
Tazkia Nizami	: 13522032
Farhan Nafis Rayhan	: 13522037

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

BAB I

Model

1.1 Implementasi KNN

KNN, atau *K-th Nearest Neighbor* adalah salah satu algoritma *supervised learning* paling sederhana dalam persoalan klasifikasi. Algoritma ini melakukan klasifikasi berdasarkan kedekatan data pada ruang *feature*. KNN bekerja dengan mengidentifikasi *k* tetangga terdekat dari sebuah data uji, kemudian menentukan kelas data tersebut berdasarkan kelas mayoritas dari tetangga-tetangga terdekatnya. Pada implementasi algoritma ini, jarak antar data diukur menggunakan salah satu dari *metric* berikut, jarak Euclidean, Manhattan, atau jarak Minkowski.

Algoritma KNN dikenal sebagai *lazy learner*. Alasannya adalah karena algoritma ini tidak membutuhkan pemodelan atau fitting. Data *training* akan disimpan sepenuhnya dalam model secara utuh. Ketika ada data baru yang ingin diprediksi, algoritma akan menghitung jarak antara data tersebut dengan setiap data dalam *training set*. Akibatnya KNN termasuk pula sebagai *Instance based classifier*. Algoritma ini tidak akan membuat hipotesis, setiap prediksi berdasarkan apa data uji saat ini. Hal ini menjadi keunggulan algoritma KNN, yaitu kesederhanaannya dan kemampuannya untuk menangani data non-linear tanpa asumsi distribusi tertentu.

Pada tugas besar ini, KNN diimplementasikan menggunakan *library pandas* serta *numpy*. Terdapat fungsi dinamis distance_function() yang implementasinya dapat berubah berdasarkan metrik jarak yang dipilih (euclidean, manhattan, minkowski). Metode fit hanya akan menyimpan seluruh *training set* ke dalam model. Selanjutnya, untuk prediksi suatu data baru dengan *predict()*, perhitungan dilakukan secara vectorized agar lebih efektif. Dengan memanfaatkan np.apply_along_axis(), algoritma dapat menghitung jarak fitur terhadap keseluruhan *training set* secara sekaligus. Kemudian hasil jarak disortir dan dipilih K tetangga teratas. Sekali lagi dengan np.apply_along_axis(), dijalankan fungsi untuk mencari kelas yang muncul paling banyak dengan np.bincount().argmax(). Akhirnya, diperoleh kelas prediksi bagi seluruh data *test set*.

1.2 Implementasi Naive Bayes

Naive Bayes merupakan metode klasifikasi yang didasarkan pada teorema Bayes. Algoritma ini bekerja dengan menghitung probabilitas suatu data termasuk dalam kelas tertentu berdasarkan nilai fitur-fiturnya, dengan asumsi bahwa fitur-fitur tersebut saling bebas. Dalam implementasi ini, diasumsikan fitur-fitur memiliki distribusi Gaussian (normal).

Proses pemodelan / Fitting dimulai dengan melihat semua data unik dari setiap class, kemudian untuk masing-masing dihitung kemungkinan munculnya class tersebut untuk menunjukkan proporsi sampel untuk setiap kelas. Kemudian untuk masing-masing feature juga dihitung rata-rata dan variansinya. Rata-rata nantinya akan digunakan sebagai parameter dalam perhitungan probabilitas gaussian. Untuk perhitungan variansi, ditambahkan tambahan nilai yang sangat kecil $(1e^{-7})$ untuk mencegah error pembagian nol ketika melakukan pembagian.

Implementasi ini mengasumsikan bahwa setiap fitur dalam data berdistribusi normal. Dengan menggunakan perhitungan probabilitas Gaussian, kita dapat mengukur seberapa mungkin suatu data berasal dari kelas tertentu berdasarkan kedekatannya dengan mean dari kelas tersebut. Semakin dekat nilai fitur dengan mean kelasnya, semakin tinggi probabilitas nya.

Pada saat memprediksi sebuah variabel target dari sebuah data, dibuatlah fungsi predict_proba(). Fungsi ini menghitung logaritma probabilitas kemiripan class tersebut sesuai dengan model yang telah dibangun. Jika data yang diukur mirip dengan target class tertentu, maka akan di-predict sebagai target class tersebut.

1.3 Implementasi ID3

Algoritma ID3 (Iterative Dichotomiser 3) adalah salah satu metode pohon keputusan yang digunakan untuk klasifikasi. Implementasi ini membangun pohon keputusan berdasarkan pengukuran information gain untuk memilih fitur terbaik yang memisahkan data ke dalam kelas-kelasnya. Algoritma ini diawali dengan proses discretization dengan mengubah data kontinu menjadi data diskrit dengan membagi nilai-nilai fitur menjadi beberapa bin menggunakan metode equal-width binning. Setelah data di-diskretisasi, algoritma menghitung entropi sebagai ukuran ketidakpastian data, kemudian menghitung information gain untuk menentukan fitur yang memberikan pemisahan terbaik. Fitur dengan information gain tertinggi dipilih sebagai node pada pohon.

Proses pembangunan pohon dilakukan secara rekursif melalui fungsi _build_tree. Fungsi ini yang membuat cabang-cabang pohon berdasarkan nilai unik dari fitur terbaik yang dipilih. Proses ini berlanjut hingga kedalaman maksimum tercapai atau tidak ada lagi fitur yang memberikan information gain signifikan. Ketika suatu cabang mencapai kondisi terminal, algoritma menetapkan kelas mayoritas dari data yang tersisa sebagai prediksi untuk cabang tersebut.

Setelah pohon dibangun, prediksi dilakukan dengan menelusuri pohon berdasarkan nilai fitur data input. Fungsi predict menerima data baru, melakukan proses

diskretisasi, dan mengarahkan setiap sampel melalui pohon untuk menentukan kelasnya. Fungsi predict_proba menghitung probabilitas prediksi untuk setiap kelas berdasarkan pohon yang telah dibangun. Untuk mengevaluasi performa model, fungsi score menghitung akurasi prediksi dengan membandingkan hasil prediksi dengan label asli.

BAB II

Data Cleaning dan Preprocessing

2.1 Tahap Cleaning

Pada tahap *cleaning*, dilakukan metode imputasi dengan nilai median pada kolom numerik dan binary, sedangkan pada kolom kategorikal dilakukan imputasi dengan metode modus.

2.1.1 Imputing Kolom Numerikal

Imputing kolom numerik dilakukan dengan nilai median karena alasan-alasan berikut:

1. Distribusi data nonsimetris: Data kolom numerik yang digunakan sangat asimetrik, terbukti dari proses *exploratory data analysis* yang menunjukkan histogram yang sangat *skewed.* Hal ini mendukung alasan untuk melakukan proses *imputing* dengan nilai median, karena tidak seperti proses *imputing* dengan rata-rata yang membutuhkan distribusi data simetris, *imputing* dengan nilai median tidak memiliki prekondisi distribusi data tertentu.
2. Ketahanan terhadap *outliers*: Proses *imputing dengan* nilai median juga cenderung lebih *robust* terhadap keberadaan *outliers*. Hal ini karena mengganti *missing values* dengan median menjaga bentuk distribusi data.

Proses *imputing* dilakukan agar data-data dengan *missing values* tidak menghambat proses *training*. Proses *imputing* ini dilakukan dengan kelas `ImputeNumerical`.

2.1.2 Imputing Kolom Kategorikal

Imputing kolom kategorikal dilakukan dengan nilai modus karena alasan-alasan berikut:

1. Meminimalkan bias terhadap nilai kolom yang tidak sering muncul: Mengganti *missing values* dengan nilai modus tidak akan menimbulkan bias terhadap nilai kolom yang jarang muncul.
2. Metode valid untuk kolom kategorikal: Ketika nilai-nilai pada kolom tidak memiliki urutan/*ordering* tertentu, mengganti *missing values* dengan nilai

modus adalah pilihan yang baik, karena pada nilai kategorikal/nonordinal, nilai lainnya seperti median dan rata-rata tidak terdefinisi.

Proses *imputing* dilakukan agar data-data dengan *missing values* tidak menghambat proses *training*. Proses *imputing* ini dilakukan dengan kelas `ImputeCategorical`.

2.2 Tahap Preprocessing

Pada tahap *preprocessing*, dilakukan *min max feature scaling* pada kolom numerikal. Setelah itu, pada kolom kategorikal dilakukan metode *one hot encoding.* Setelah itu, dilakukan *max absolute scaling* pada kolom kategorikal yang telah di-*one-hot encoded*. Terakhir, dilakukan pemilihan K kolom yang merepresentasikan data terbaik dengan fungsi SelectKBest.

2.2.1 Min Max Scaling pada Kolom Numerikal

Pada tahap ini, dilakukan min max scaling pada kolom numerikal. Proses scaling ini dilakukan karena:

1. Rentang nilai yang sama: Melakukan Min Max Scaling membuat seluruh nilai kolom berada pada rentang yang sama, yang dalam hal ini adalah rentang [0, 1].
2. Memastikan performa algoritma yang sensitif terhadap nilai: Algoritma seperti KNN yang sangat sensitif terhadap perhitungan jarak sangat sensitif terhadap nilai-nilai besar. Dengan melakukan *min max scaling*, semua fitur akan berkontribusi sama.

2.2.2 One Hot Encoding pada Kolom Kategorikal

Pada tahap ini dilakukan *one hot encoding* pada kolom kategorikal. Tahap *one hot encoding* dilakukan karena:

1. Mengubah nilai kategorik menjadi numerik: Banyak algoritma *machine learning* membutuhkan fitur numerikal. Tahap *one-hot encoding* mengubah nilai-nilai pada kolom kategorik menjadi numerik dengan tetap mempertahankan keunikan nilai.
2. Mencegah kesalahan pada interpretasi nilai ordinal: One hot encoding menjaga independensi antar nilai kategorikal, dan mencegah kesalahan interpretasi nilai ordinal.

2.2.3 Max Absolute Scaling pada Kolom Kategorikal

Pada tahap ini dilakukan *max absolute scaling* pada kolom kategorikal. Tahap ini dilakukan karena:

1. Menjaga *sparsity* data: Teknik ini menjaga *sparsity* data. Karena kolom yang diterapkan teknik ini adalah kolom *one-hot* yang cenderung *sparse*, maka teknik ini cocok digunakan.
2. Menjaga relasi/hubungan antar fitur: Teknik ini tidak membuat fitur menjadi sebuah rentang tertentu, yang berarti hubungan antar fitur tetap terjaga.

2.2.4 Pemilihan K Kolom Terbaik

Tahap terakhir dalam proses *data processing* adalah dengan memilih K kolom terbaik yang merepresentasikan keseluruhan data. Tahap ini dilakukan karena:

1. Meningkatkan performa model: Dengan memilih K fitur/kolom terbaik, *noise* atau *irrelevant data* dapat dikurangi, yang kemudian akan meningkatkan performa model.
2. Mengurangi *overfitting*: Dengan hanya menggunakan K fitur yang terbaik, proses ini akan membantu model men-generalisasi *training* data.

BAB III

Perbandingan Hasil Prediksi

3.1 Perbandingan KNN

```
Cross validation scores: [0.75102477 0.75444662 0.75083767 0.75240607 0.74869894]
Mean accuracy: 0.7514828154887327
Standard deviation of accuracy: 0.001898027937470824
F1 score: 0.4487272891720095
```

	attack_cat	predicted
15482	Generic	Generic
133349	Generic	Generic
80485	Exploits	DoS
29972	DoS	Exploits
18339	Normal	Normal

Dari hasil perbandingan di atas dapat dilihat bahwa model yang diimplementasikan secara manual memiliki performa lebih buruk dibandingkan model dari *library*.

Kami juga menemukan bahwa model ini sangat sensitif terhadap *scaling*, karena menggunakan *min max scaling* pada fitur numerikal dan *max absolute scaling* pada fitur kategorikal membantu performa model dengan signifikan, terlihat dari F1 score validasi yang meningkat.

3.2 Perbandingan Naive Bayes

Berikut adalah hasil dari prediksi data dengan menggunakan library bawaan dari sklearn.

```
Cross validation scores: [0.45236143 0.44516129 0.44699508 0.44834961 0.42942183]
Mean accuracy: 0.44445784898607527
Standard deviation of accuracy: 0.007881963163590695
F1 score: 0.2066428837182126
```

	attack_cat	predicted
15482	Generic	Generic
133349	Generic	Generic
80485	Exploits	Analysis
29972	DoS	Backdoor
18339	Normal	Normal

Berikut adalah hasil dari prediksi data dengan menggunakan class yang kami buat dengan bantuan dari library numpy.

```
Cross validation scores: [0.30033862 0.28173231 0.28612676 0.28327511 0.27005062]
Mean accuracy: 0.28430468331837105
Standard deviation of accuracy: 0.009708373912064333
F1 score: 0.13810860769266328
```

	attack_cat	predicted
15482	Generic	Generic
133349	Generic	Shellcode
80485	Exploits	Shellcode
29972	DoS	Shellcode
18339	Normal	Worms

Dari hasil perbandingan di atas, dapat dilihat bahwa nilai F1 score dari implementasi F1-score manual lebih buruk dibandingkan implementasi F1-score dari library. Hal ini karena implementasi library sudah di optimasi menggunakan teknik yang lebih *advanced.*

Dari hasil eksperimentasi kami, kami juga menemukan bahwa *missing values* lebih baik diganti menggunakan teknik *imputing* dibandingkan di-drop. Hal ini karena banyak row yang memiliki *missing values* cukup banyak, kurang lebih ⅞ dari total data. Hal ini tentu akan memperkecil data *training* secara signifikan.

3.3. Perbandingan ID3

Berikut adalah hasil dari prediksi data oleh model ID3 dengan menggunakan library bawaan dari sklearn.

```
Cross validation scores: [0.78417394 0.78795224 0.78391673 0.78904969 0.77860555
Mean accuracy: 0.7847396298126419
Standard deviation of accuracy: 0.003674822461361534
F1 score: 0.5539569171911204
```

	attack_cat	predicted
15482	Generic	Generic
133349	Generic	Generic
80485	Exploits	DoS
29972	DoS	Fuzzers
18339	Normal	Normal

Berikut adalah hasil dari prediksi data dengan menggunakan class yang kami buat dengan bantuan dari library numpy.

```
[Parallel(n_jobs=10)]: Using backend LokyBackend with 10 concurrent workers.
[Parallel(n_jobs=10)]: Done   5 out of   5 | elapsed:  1.1min finished
Cross validation scores: [0.73794333 0.73755124 0.73718543 0.74160548 0.73533186]
Mean accuracy: 0.7379234654982231
Standard deviation of accuracy: 0.002047303627154922
F1 score: 0.4480867130136149
```

	attack_cat	predicted
15482	Generic	Generic
133349	Generic	Generic
80485	Exploits	Exploits
29972	DoS	DoS
18339	Normal	Normal

Sama seperti Naive Bayes, model yang diimplementasikan sendiri untuk ID3 memiliki performa lebih buruk dibandingkan model yang menggunakan *library*.

Selain itu, kami juga menemukan bahwa proses *imputing* meningkatkan performa model. Hal ini karena dengan *imputing* dan tidak men-*drop* row data, data *training* yang model dapatkan semakin banyak, memungkinkan proses *training* yang lebih relevan ke data aslinya.

Selain itu, *feature scaling* juga memperbaiki performa model, tepatnya Min Max Absolute Scaling pada kolom numerikal. Hal ini karena dengan *scaling*, didapat representasi fitur yang lebih *uniform*, dibanding tanpa *scaling* dimana setiap fitur memiliki rentangnya masing-masing.

Kontribusi

Nama	NIM	Pembagian Tugas
Ahmad Naufal Ramadan	13522005	Preprocessing & Implementasi ID3
Kristo Anugrah	13522024	Preprocessing
Tazkia Nizami	13522032	Implementasi Naive-Bayes
Farhan Nafis Rayhan	13522037	Implementasi KNN & Laporan

Referensi

- Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
- Slide Presentasi Kuliah Inteligensi Buatan