IF3170 Inteligensi Buatan

IMPLEMENTASI ALGORITMA K-NEAREST NEIGHBOUR DAN NAIVE-BAYES DALAM MODEL KLASIFIKASI

Laporan Tugas Besar 2

Disusun untuk memenuhi tugas mata kuliah IF3170 Inteligensi Buatan pada Semester 1 (satu) Tahun Akademik 2023/2024

Oleh

Michael Jonathan Halim	13521124
Chiquita Ahsanunnisa	13521129
Enrique Alifio Ditya	13521142
Rava Maulana	13521149

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2023

DAFTAR ISI

DAFTAR ISI **1**

BAB I
DATA UNDERSTANDING **2**

1.1 Latar Belakang 2

1.2 Deskripsi Data 2

1.3 Exploratory Data Analysis 3

1.3.1 Duplicate dan Missing Value 3

1.3.2 Outlier 4

1.3.3 Distribusi Kelas Fitur Target 4

1.3.4 Korelasi Terhadap Target 5

BAB II
DATA CLEANING DAN PREPROCESSING **7**

2.1. Data Cleaning 7

2.2. Feature Engineering 7

2.3. Feature Selection 8

2.4. Feature Scaling 9

2.5. Principal Component Analysis (PCA) 9

BAB III
IMPLEMENTASI MODEL **11**

3.1. K-Nearest Neighbor (K-NN) 11

3.2. Naive-Bayes 13

BAB IV
EVALUASI **16**

4.1. Hasil Klasifikasi 16

4.2. Confusion Matrix 19

4.3. Perbandingan dengan Pustaka Scikit Learn 20

LAMPIRAN **22**

Lampiran 1 Pranala GitHub Repository 22

DAFTAR PUSTAKA **23**

BAB I

DATA UNDERSTANDING

1.1 Latar Belakang

Ponsel hadir dengan berbagai macam harga, fitur, spesifikasi, dan lainnya. Estimasi dan prediksi harga adalah bagian penting dari strategi konsumen. Memutuskan harga yang tepat untuk sebuah produk sangat penting untuk kesuksesan pasar sebuah produk. Sebuah produk baru yang akan diluncurkan, harus memiliki harga yang tepat agar konsumen merasa pantas untuk membeli produk tersebut. Pada laporan ini, akan dipaparkan tahapan memprediksi tingkat kisaran harga ponsel diberikan data informasi mengenai fitur-fitur handphone dan spesifikasinya.

1.2 Deskripsi Data

Dataset terdiri atas 21 fitur, dengan rincian 7 fitur kategorikal dan 14 fitur numerik. Berikut merupakan rincian dari setiap fitur yang terdapat pada dataset.

Tabel 1.1 Deskripsi dan Jenis Fitur

Fitur	Jenis	Deskripsi
battery_power	Numerik	Total energi baterai dalam satu waktu diukur dalam mAh
blue	Kategorikal Biner	Memiliki bluetooth atau tidak
clock_speed	Numerik	Kecepatan mikroprosesor menjalankan instruksi
dual_sim	Kategorikal Biner	Memiliki dukungan dual sim atau tidak
fc	Numerik	Resolusi kamera depan dalam megapiksel
four_g	Kategorikal Biner	Memiliki 4G atau tidak
int_memory	Numerik	Memori internal dalam gigabyte
m_dep	Numerik	Ketebalan ponsel dalam cm
mobile_wt	Numerik	Berat ponsel
n_cores	Numerik	Jumlah core prosesor

pc	Numerik	Resolusi kamera utama dalam megapiksel
px_height	Numerik	Tinggi resolusi piksel
px_width	Numerik	Lebar resolusi piksel
ram	Numerik	Ukuran RAM dalam megabyte
sc_h	Numerik	Tinggi layar ponsel dalam cm
sc_w	Numerik	Lebar layar ponsel dalam cm
talk_time	Numerik	Waktu telepon maksimum dalam satu kali pengisian baterai
three_g	Kategorikal Biner	Memiliki 3G atau tidak
touch_screen	Kategorikal Biner	Memiliki layar sentuh atau tidak
wifi	Kategorikal Biner	Memiliki wifi atau tidak
price_range (target)	Kategorikal Multi Kelas (4 kelas)	Rentang harga dengan nilai 0 (biaya rendah), 1 (biaya sedang), 2 (biaya tinggi) atau 3 (biaya sangat tinggi)

1.3 *Exploratory Data Analysis*

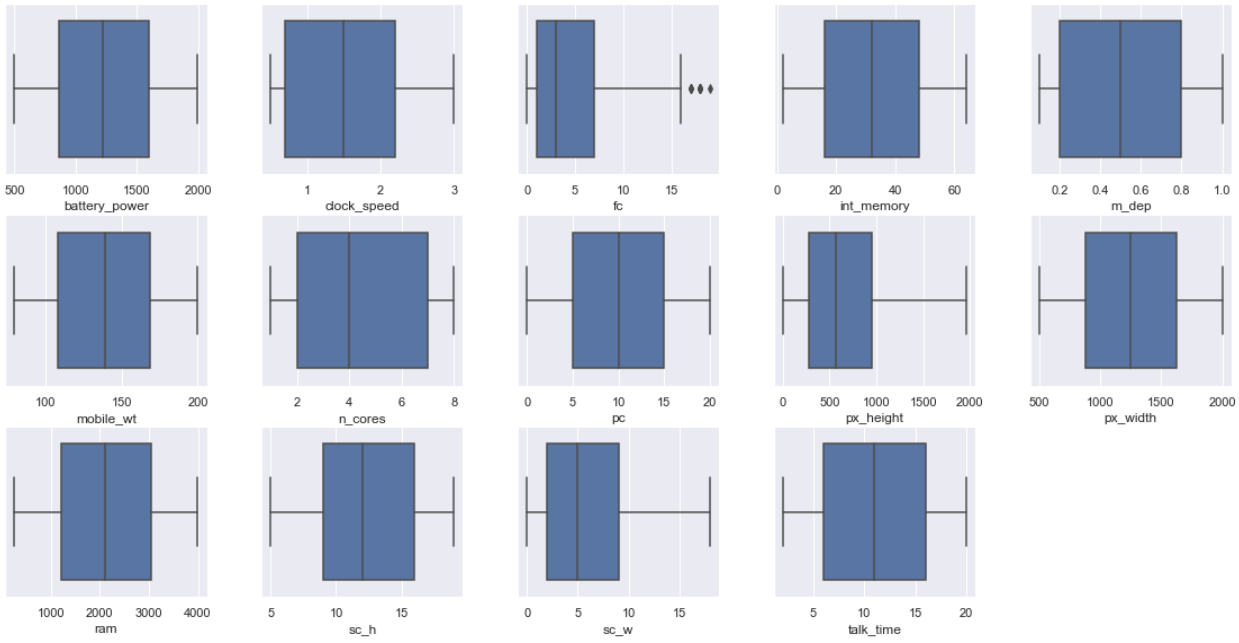
Tahap *exploratory data analysis* (EDA) dilakukan untuk mendapatkan wawasan lebih lanjut terhadap data. Wawasan ini kemudian digunakan untuk menentukan teknik data *cleaning* dan *preprocessing* yang akan digunakan.

1.3.1 *Duplicate* dan *Missing Value*

Berdasarkan hasil analisis, didapat bahwa tidak ada data yang terduplikasi pada dataset, sehingga tidak perlu dilakukan pembersihan data duplikat. Berdasarkan hasil analisis, didapat bahwa dataset terdiri atas 1400 baris dengan setiap fiturnya tidak ada yang bernilai Null. Artinya, tidak ada data dengan nilai yang hilang (*missing value*) pada dataset tersebut, sehingga tidak perlu dilakukan perbaikan atau pembersihan data akibat *missing value*.

1.3.2 *Outlier*

Metode identifikasi *outlier* yang digunakan ada dua, yaitu metode IQR dan standar deviasi (digunakan tiga kali standar deviasi dari mean sebagai batas). Analisis pencilan hanya dilakukan untuk fitur numerik. Dengan menggunakan metode IQR dan standar deviasi, fitur yang memiliki pencilan hanyalah fc. Metode IQR mengidentifikasi 11 pencilan dari fitur tersebut, sementara metode standar deviasi mengidentifikasi 8 pencilan. *Box plot* dari setiap fitur numerik beserta pencilannya (jika ada) ditunjukkan pada Gambar 1.1. Jumlah pencilan yang relatif sedikit menandakan bahwa tidak perlu dilakukan penanganan nilai pencilan pada data.



Gambar 1.1 *Box Plot* Untuk Setiap Fitur Numerik

(Sumber: Dokumentasi Penulis)

1.3.3 Distribusi Kelas Fitur Target

Tabel 1.2 menunjukkan statistik dasar untuk fitur target, yaitu price_range. Nilai dari kelas 1 adalah biaya rendah (dilambangkan dengan angka 0 pada data). Nilai dari kelas 2 adalah biaya sedang (dilambangkan dengan angka 1 pada data). Nilai dari kelas 3 adalah biaya tinggi

(dilambangkan dengan angka 2 pada data). Nilai dari kelas 4 adalah biaya sangat tinggi (dilambangkan dengan angka 3 pada data).

Tabel 1.2 Distribusi Dari Setiap Kelas Untuk Fitur Target

Fitur	Proporsi Kelas 1 (%)	Proporsi Kelas 2 (%)	Proporsi Kelas 3 (%)	Proporsi Kelas 4 (%)	Modus
price_range	25.57	25.43	24.64	24,36	0

Pada Tabel 1.2, dapat dilihat bahwa fitur target memiliki pembagian proporsi yang sama rata untuk setiap kategori. Hal tersebut menandakan bahwa tidak diperlukan penanganan khusus untuk *imbalanced data* ketika akan membuat model. Selain itu, model yang dapat dibuat dari dataset ini akan mampu melakukan generalisasi dengan lebih baik karena dilatih pada jumlah sampel yang cenderung sama untuk setiap kategori pada fitur target.

1.3.4 Korelasi Terhadap Target

Korelasi setiap fitur dengan fitur target dihitung menggunakan koefisien korelasi *rank-biserial* dan Kendall. Korelasi antara fitur kategorikal dan target dihitung menggunakan koefisien korelasi rank-biserial, sementara korelasi antara fitur numerik dengan target dihitung menggunakan koefisien korelasi Kendall. Tabel 1.3 dan 1.4 menunjukkan hasil koefisien korelasi untuk setiap fitur (metode perhitungan disesuaikan dengan jenis dari setiap fitur). Semakin besar koefisien korelasi dari sebuah fitur, semakin kuat hubungan antara fitur tersebut dengan fitur target.

Tabel 1.3 Tabel Koefisien Korelasi Fitur Kategorikal

Fitur	Koefisien Rank-Biserial
blue	0.00013
dual_sim	-0.00003
four_g	0.00000
three_g	0.00010

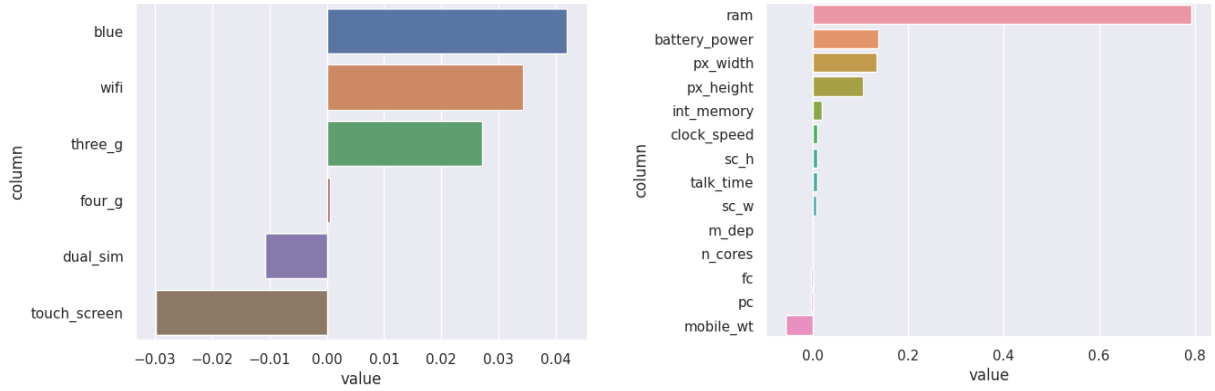
Tabel 1.4 Tabel Koefisien Korelasi Fitur Numerik

Fitur	Koefisien Kendall
battery_power	0.1375
clock_speed	0.00986
fc	-0.00421
int_memory	0.01883

touch_screen	-0.00010
wifi	0.00011

m_dep	0.00072
mobile_wt	-0.0558
n_cores	-0.00032
pc	-0.00456
px_height	0.10558
px_width	0.13328
ram	0.79345
sc_h	0.00986
sc_w	0.00674
talk_time	0.00868

Gambar 1.2 menunjukkan nilai koefisien korelasi setiap fitur dengan fitur target. *Bar plot* digunakan karena dapat menunjukkan fitur-fitur yang berkorelasi tinggi dengan target secara lebih jelas. Tiga fitur dengan korelasi tertinggi adalah ram, battery_power, dan px_width. Fitur-fitur ini berpotensi untuk meningkatkan kinerja model dan dapat dikembangkan lebih lanjut sehingga diperoleh fitur-fitur turunan dari fitur tersebut. Fitur yang berkorelasi rendah, seperti four_g, m_dep, dan n_cores dapat dihilangkan untuk meningkatkan efisiensi pelatihan model (teknik filtering).



Gambar 1.2 Bar Plot Perbandingan Korelasi Setiap Fitur dengan Fitur Target

(Sumber: Dokumentasi Penulis)

BAB II

DATA CLEANING DAN *PREPROCESSING*

2.1. *Data Cleaning*

Berdasarkan analisis, mayoritas data dapat dikatakan sudah cukup bersih, kecuali beberapa fitur yang mengandung nilai noise. Contoh dari kasus ini antara lain baris dengan fitur `px_height` atau `sc_w` mendekati nol. Berdasarkan riset sederhana, ditemukan bahwa:

- Beberapa ponsel yang paling awal tersedia secara komersial, seperti Motorola DynaTAC 8000X, memiliki resolusi layar hanya 5 piksel.
- Beberapa ponsel paling awal dengan layar memiliki lebar sekitar 2-3 sentimeter.

Karena itu, untuk ponsel dengan fitur-fitur tersebut yang memiliki nilai di bawah ambang kewajaran (resolusi kurang dari 5 piksel atau lebar ponsel kurang dari 2 sentimeter), dapat dianggap sebagai outlier atau kesalahan data, atas dasar *common knowledge*.

Berbagai upaya telah dilakukan untuk mengatasi ini, salah satunya dengan memaksa nilai `px_height` atau `sc_w` yang mendekati nol menjadi null kemudian diimputasi dengan rata-rata dan algoritma K-NN Imputer. Namun, metode ini memperburuk performa validasi model sekitar 2% untuk metrik akurasi, sehingga data dibiarkan seadanya.

2.2. *Feature Engineering*

Dari berbagai eksperimen dengan metode iteratif, diidentifikasi tiga fitur baru, yakni `screen_size`, `battery_ram_sum`, dan `ram_mobile_weight` meningkatkan skor validasi model. Fitur `screen_size`, yang dihitung sebagai hasil kali antara tinggi layar (`sc_h`) dan lebar layar (`sc_w`), menunjukkan korelasi positif yang moderat (0,0412) dengan fitur target (`price_range`). Di sisi lain, `battery_ram_sum`, yang mewakili jumlah daya baterai dan RAM, menunjukkan korelasi positif yang sangat tinggi (0,9255) dengan target, yang menunjukkan bahwa perangkat dengan gabungan daya baterai dan RAM yang lebih besar cenderung memiliki kisaran harga yang lebih tinggi. Demikian pula, `ram_mobile_weight`, yang dihitung sebagai kontras (selisih)

antara RAM dan berat ponsel, menunjukkan korelasi positif yang kuat (0,9175) dengan target, yang menunjukkan bahwa perangkat dengan RAM yang lebih besar dibandingkan dengan beratnya dikaitkan dengan kisaran harga yang lebih tinggi. Temuan ini juga menunjukkan fitur-fitur yang direkayasa menangkap pola yang berarti dalam data, memberikan informasi yang berharga kepada model untuk meningkatkan kinerja prediktifnya. Korelasi yang kuat menunjukkan bahwa fitur-fitur ini sangat mengindikasikan variabel target, memberikan wawasan yang berharga untuk memprediksi harga ponsel secara akurat.

2.3. *Feature Selection*

Dalam fase pengembangan model, ditemukan hal penting berikut: *dropping* fitur kategorikal bersama dengan fitur RAM meningkatkan kinerja model pada data validasi, terutama ketika menggunakan algoritma K-Nearest Neighbors (K-NN). Peningkatan ini menunjukkan bahwa K-NN sebagai algoritma berbasis jarak, mungkin kesulitan dengan sifat inheren dari variabel kategorikal. Metrik jarak yang digunakan dalam K-NN bergantung pada kedekatan numerik titik data, dan fitur kategorikal sering kali tidak memiliki representasi numerik yang berarti. Pengecualian kolom kategorikal ini kemungkinan besar dapat mengurangi noise yang disebabkan oleh nilai non-numerik, sehingga memungkinkan model untuk melihat pola dan hubungan yang lebih baik di dalam kumpulan data.

Untuk fitur RAM, meskipun pada awalnya menunjukkan korelasi yang tinggi (sekitar 0,9) dengan variabel target, kinerja model meningkat setelah mengecualikannya dari fitur input. Hasil yang tidak terduga ini mungkin disebabkan tahap *Feature Engineering* sebelumnya, khususnya pembuatan fitur komposit yang menggabungkan RAM dengan atribut lain. Perekayasaan fitur baru telah menyebabkan kolinearitas atau noise pada dataset, yang memengaruhi kinerja model.

Dengan menghilangkan kolom kategorikal dan fitur RAM, model menjadi lebih *robust* dan tidak terlalu sensitif terhadap informasi yang tidak relevan atau berlebihan, sehingga menghasilkan akurasi prediksi yang lebih baik pada data validasi. Temuan ini menggarisbawahi pentingnya

pemilihan fitur dan *preprocessing* yang cermat dalam mengoptimalkan kinerja model pembelajaran mesin, bahkan ketika dihadapkan pada hasil yang tampaknya berlawanan dengan intuisi.

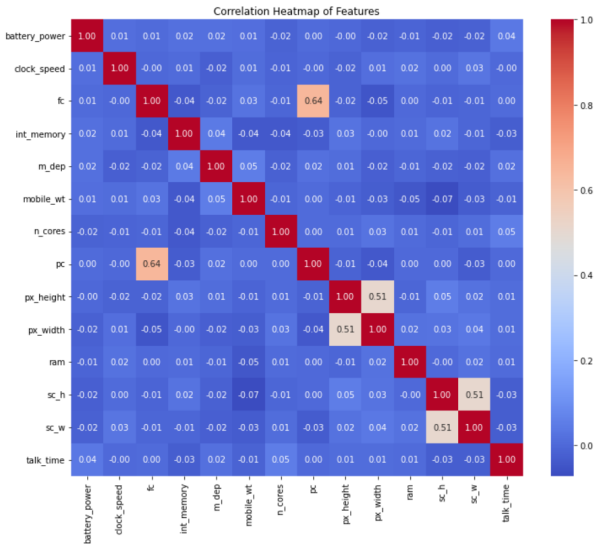
2.4. *Feature Scaling*

Untuk menggunakan model berbasis distance, kombinasi linier, hyperplane, atau model lainnya yang sensitif terhadap skala dari tiap fitur, pada umumnya perlu dilakukan langkah penskalaan. Dengan mempertimbangkan distribusi mayoritas fitur, digunakanlah metode standardisasi untuk fitur-fitur numerik. Hal ini disebabkan metode mayor lainnya dalam penskalaan, yakni Min-max scaling, biasanya direkomendasikan untuk distribusi normal, namun sesuai analisis pada Soal 5, dapat dilihat mayoritas fitur tidak sepenuhnya dikatakan terdistribusi normal.

Meskipun itu, ditemukan bahwa *scaling* justru menurunkan performa model pada data validasi. Hal ini mungkin disebabkan fitur menjadi kehilangan bobot kepentingannya. Karena itu, tahap ini tidak diterapkan pada *pipeline preprocessing* akhir.

2.5. *Principal Component Analysis* (PCA)

Setelah menggali lebih dalam data latih, sebuah pengamatan kritis muncul: multikolinieritas di antara kolom-kolom tertentu berdampak buruk pada kinerja model. Untuk mengatasi masalah ini, Singular Value Decomposition (SVD) dengan Principal Component Analysis (PCA) diterapkan. Dasar pemikiran di balik penggunaan PCA terletak pada kemampuannya untuk mengubah fitur-fitur yang berkorelasi menjadi sekumpulan variabel yang tidak berkorelasi linier *(principal components)*.



Gambar 2.1 Korelasi antar variabel sebelum PCA

(Sumber: Dokumentasi Penulis)

PCA secara desain menghasilkan komponen-komponen utama yang bebas linier. Dengan mengurangi dimensionalitas sambil mempertahankan aspek yang paling informatif dari data, PCA tidak hanya mengurangi efek buruk dari multikolinieritas tetapi juga membantu Naive-Bayes dalam menegakkan asumsi *naive* dasarnya, yakni setiap variabel input bersifat independen terhadap satu sama lain.

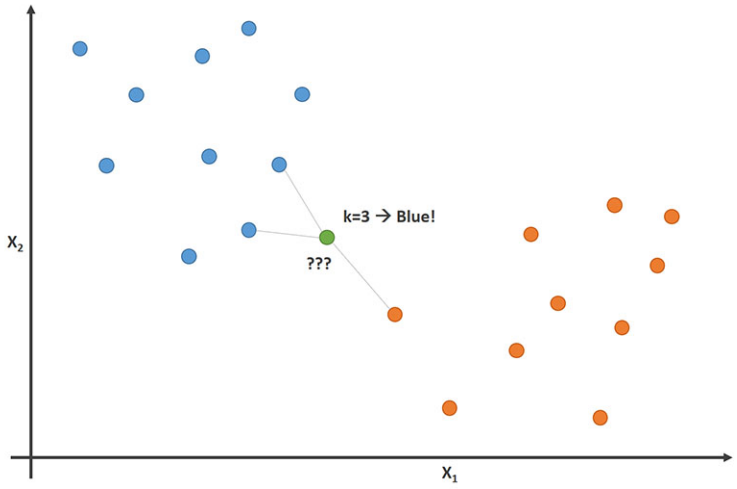
Langkah ini terbukti sangat penting dalam meningkatkan kemampuan generalisasi model dan, akibatnya, akurasi prediktif pada data baru yang belum pernah dilihat sebelumnya. Eksperimen menunjukkan penerapan PCA dengan tujuh komponen menghasilkan kinerja yang optimal pada data validasi. Tahap ini menggarisbawahi keseimbangan yang rumit antara pengurangan dimensi dan retensi informasi, menekankan pentingnya rekayasa fitur yang bijak dalam mengejar optimasi model.

BAB III

IMPLEMENTASI MODEL

3.1. K-Nearest Neighbor (K-NN)

K-Nearest Neighbor (K-NN) merupakan jenis model pembelajaran mesin yang digunakan untuk jenis *task supervised learning*. Model K-NN bekerja dengan mengidentifikasi 'k' data *point* yang berdekatan dengan data *point* yang ingin diprediksi. Jarak antar data *point* dapat dihitung dengan berbagai cara, diantaranya adalah *euclidean distance* dan *manhattan distance*. Setelah diidentifikasi k data *point* terdekat, kelas mayoritas dari k data *point* tersebut akan dijadikan sebagai kelas prediksi dari data *point* yang ingin diprediksi.



Gambar 3.1 Ilustrasi Cara Kerja Model K-Nearest Neighbor

(Sumber: https://rapidminer.com)

Gambar 3.1 menunjukkan cara kerja model K-NN dalam melakukan prediksi. Data *point* berwarna biru dan oranye merupakan data *train* yang akan disimpan pada tahap pelatihan model. Hal tersebut membuat model K-NN disebut sebagai "*lazy learner*" karena tidak memiliki tahap pelatihan dan sebagian besar perhitungan dilakukan pada tahap prediksi. Data *point* berwarna hijau merupakan data yang ingin diprediksi kelasnya. Jika digunakan k = 3, seperti pada gambar, didapat tiga data *point* terdekat memiliki kelas biru, biru, dan oranye. Karena

kelas biru merupakan kelas mayoritas, model K-NN akan memprediksi data *point* baru (hijau) memiliki kelas biru.

Model K-NN yang dibuat diimplementasikan menggunakan bahasa pemrograman Python sebagai sebuah kelas dengan atribut dan metode tertera pada Tabel 3.1.

Tabel 3.1 Deskripsi kelas algoritma K-NN

Atribut	
`k`	Jumlah *neighbor* untuk penentuan klasifikasi
`weights`	(‘uniform’ atau ‘distance’) Metode pembobotan untuk setiap *neighbor*. Metode ‘distance’ akan memberikan bobot yang lebih besar untuk data *point* yang lebih dekat, sementara metode ‘uniform’ akan memberikan bobot yang seragam.
`p`	Parameter untuk metrik Minkowski (jika digunakan *minkowski distance*).
`metric`	Digunakan untuk menentukan fungsi perhitungan jarak antara *euclidean distance, manhattan distance,* atau *minkowski distance*).
`n_jobs`	Jumlah *task* parallel yang akan dikerjakan selama menghitung jarak antar-data.
`verbose`	Digunakan jika ingin menampilkan *log* di terminal selama proses pelatihan dan prediksi model.
`X_train`	Matriks yang menyimpan seluruh variabel prediktor dari data *train*
`y_train`	*Array* yang menyimpan label dari data *train*
Metode	
`KNN(k: int, n_jobs: int, metric: string, p: int, weights: string, verbose: boolean) → void`	
Metode ini merupakan konstruktor model.	
`_get_nearest_neighbours(test: matrix) → list`	
Fungsi pencarian *neighbor* yang menerima masukan sebuah matriks yang berisi kumpulan data *point* yang ingin diprediksi dan mengembalikan daftar *neighbor* dari kumpulan data *point* tersebut.	
`fit(X_train: matriks, y_train: array) → void`	
Fungsi pelatihan model yang menerima masukan data *train* beserta labelnya dan menyimpan	

data-data tersebut.
_predict_instance(row: array) → int
Fungsi prediksi yang menerima masukan sebuah data *point* dan mengembalikan prediksi kelas untuk data tersebut.
predict(X_test: matriks) → array
Fungsi prediksi yang menerima masukan data *test* dan mengembalikan *array* yang berisi label-label hasil prediksi untuk data *test* tersebut.

3.2. Naive-Bayes

Salah satu model yang dapat digunakan sebagai *classifier* adalah model Naive-Bayes. Model ini adalah *probabilistic classifier* yang didasarkan pada Teorema Bayes. Teorema Bayes adalah teorema yang digunakan untuk menghitung probabilitas *conditional*. Teorema tersebut memiliki formula sebagai berikut.

$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$, dengan $P(A \mid B)$ adalah probabilitas A jika diberikan (*given*) kondisi B.

Model Naive-Bayes mengasumsikan bahwa setiap fitur *predictor* bersifat independen terhadap fitur *predictor* lainnya.

Misalkan v_j melambangkan kelas berlabel j dan a_i melambangkan nilai fitur *predictor* i, maka langkah yang dilakukan pada algoritma Naive-Bayes adalah sebagai berikut.

(1) Membuat tabel frekuensi untuk setiap fitur *predictor* untuk setiap kelas, serta tabel frekuensi untuk setiap kelas. Hal ini dilakukan untuk mempermudah perhitungan yang dilakukan
(2) Menghitung *prior probability* untuk setiap kelas label ($P(v_j)$).
(3) Menghitung *likelihood probability* dengan setiap fitur untuk setiap kelas ($P(a_i \mid v_j)$).
(4) Menghitung *posterior probability* ($P(v_j \mid a_i)$) dengan memanfaatkan hasil perhitungan pada (2) dan (3) berdasarkan Teorema Bayes.

(5) Menghitung probabilitas kelas jika diberikan (*given*) nilai-nilai fitur *predictor* ($P(v_i \mid a_1, a_2, ..., a_n)$). Kelas dengan nilai probabilitas yang paling besar adalah nilai hasil prediksinya.

Pada implementasinya, probabilitas dihitung berdasarkan probabilitas *gaussian* (distribusi normal). Hal ini diputuskan karena fitur *predictor* yang berpengaruh secara signifikan terhadap model adalah fitur numerik yang bernilai kontinu. Selain itu, sebelum model digunakan, telah dilakukan *feature engineering* sehingga fitur yang dipilih sesuai dengan asumsi pada model Naive-Bayes.

Berikut merupakan penjelasan singkat tentang implementasi algoritma Naive-Bayes yang dibuat. Algoritma ini diimplementasikan pada kelas `NaiveBayes` yang terletak pada *file* `src/lib/bayes.py`.

Tabel 3.2 Deskripsi kelas algoritma Naive Bayes

Atribut	
`class_probabilities`	Probabilitas per label. Dinyatakan dalam bentuk *dictionary* pasangan label dan nilai probabilitasnya. Berikut bentuknya.
`mean`	Nilai mean setiap fitur per labelnya. Dinyatakan dalam bentuk *dictionary* pasangan label dan mean setiap fitur.
`variance`	Nilai variansi setiap fitur per labelnya. Dinyatakan dalam bentuk *dictionary* pasangan label dan variansi setiap fitur.
Metode	
`NaiveBayes() → void`	
Metode ini merupakan konstruktor objek yang menginisiasi seluruh atribut dengan nilai None.	
`_calculate_class_probabilities(y: array) → dictionary`	
Metode ini menerima argumen y yang merupakan nilai fitur target untuk seluruh baris. Metode ini mengembalikan nilai probabilitas untuk setiap label dalam bentuk *dictionary*.	
`_calculate_mean_and_variance(X: matrix, y: array) → dictionary`	
Metode ini menerima argumen X yang merupakan matriks yang berisi nilai fitur *predictor* untuk seluruh baris dan argumen y yang merupakan nilai fitur target untuk seluruh baris. Metode ini mengembalikan nilai *mean* dan *variance* untuk setiap label dalam bentuk *dictionary*.	

fit(X: matrix, y: array) → void

Metode ini menerima argumen X yang merupakan matriks yang berisi nilai fitur *predictor* untuk seluruh baris dan argumen y yang merupakan nilai fitur target untuk seluruh baris. Metode ini mengeset atribut class_probabilities, mean, dan variance dengan nilai yang bersesuaian.

_gaussian_probability(x: float, mean: float, variance: float) → float

Metode ini menerima argumen x yang merupakan matriks yang berisi nilai fitur *predictor*, mean yang merupakan mean untuk label yang diinginkan pada fitur yang diinginkan, dan variance yang merupakan nilai variance untuk label yang diinginkan pada fitur yang diinginkan. Metode ini mengembalikan nilai probabilitas gaussian yang dihitung berdasarkan rumus berikut.

$$P = \frac{1}{\sqrt{2\pi \cdot variance}} exp(-\frac{(x-mean)^2}{2 \cdot variance})$$

_calculate_class_probabilities_given_features(features: array, label: int) → float

Metode ini menerima argumen probabilitas label jika diberikan nilai fitur *predictor* berdasarkan aturan Bayes. Berikut merupakan rumus yang digunakan dalam perhitungan pada metode ini.

$$P(label \mid f_1, f_2, ..., f_n) = P(label) \cdot \Pi_i P(f_i \mid label)$$

predict(X: matrix) → array

Metode ini menerima argumen X yang merupakan matriks yang berisi nilai fitur *predictor* untuk seluruh baris. Hasil yang dikembalikan adalah prediksi label menggunakan algoritma Naive Bayes untuk setiap baris pada X.

save(path: string) → void

Metode ini digunakan untuk menyimpan model hasil *learning* algoritma Naive Bayes ke path yang dispesifikasikan pada argumen.

static load(path: string) → object (from bytes)

Metode ini adalah metode statik yang digunakan untuk me-*load* model hasil *learning* algoritma Naive Bayes dari path yang dispesifikasikan pada argumen.

BAB IV

EVALUASI

4.1. Hasil Klasifikasi

Laporan evaluasi memberikan wawasan mengenai kinerja model K-NN dan Naive-Bayes pada data pelatihan dan validasi.

Untuk model K-NN, data pelatihan menunjukkan presisi, recall, dan F1-score yang tinggi di semua kelas (0 hingga 3), mencapai akurasi keseluruhan 96%. Secara khusus, kelas 0 menunjukkan kinerja yang luar biasa dengan presisi 97% dan recall 99%. Set validasi mempertahankan kinerja yang kuat, meskipun dengan akurasi yang sedikit lebih rendah yaitu 93%. Model ini berkinerja baik di semua kelas, menunjukkan kemampuannya untuk menggeneralisasi data yang tidak terlihat. Metrik presisi dan recall yang seimbang menunjukkan bahwa model ini menghindari bias terhadap kelas tertentu. Tabel 4.1 dan 4.2 menunjukkan performa K-NN pada data latih dan validasi.

Tabel 4.1 Performa algoritma K-NN pada data latih

Label	**Precision**	**Recall**	**F1-score**	**Support**
0	0.97	0.99	0.98	358
1	0.95	0.96	0.96	356
2	0.94	0.94	0.94	345
3	0.98	0.95	0.96	341
Accuracy	0.96			1400
Macro Avg	0.96	0.96	0.96	1400
Weighted Avg	0.96	0.96	0.96	1400

Tabel 4.2 Performa algoritma K-NN pada data validasi

Label	**Precision**	**Recall**	**F1-score**	**Support**

0	0.94	0.96	0.95	142
1	0.90	0.90	0.90	144
2	0.92	0.90	0.91	155
3	0.97	0.96	0.96	159
Accuracy	0.93			600
Macro Avg	0.93	0.93	0.93	600
Weighted Avg	0.93	0.93	0.93	600

Sebaliknya, model Naive-Bayes menunjukkan akurasi yang lebih rendah pada data pelatihan dan validasi, yakni 85%. Meskipun model ini berkinerja cukup baik dalam mengklasifikasikan kelas 3, model ini mengalami kesulitan dalam hal presisi dan recall di kelas 1 dan 2. Hal ini menunjukkan bahwa Naive-Bayes mungkin tidak dapat menangkap kompleksitas data yang mendasari seefektif K-NN. *Trade-off* presisi-recall dalam Naive-Bayes terlihat jelas, dengan presisi yang lebih tinggi untuk kelas 3 tetapi recall yang lebih rendah, dan sebaliknya untuk kelas 1. Akurasi keseluruhan model sebesar 85% pada set validasi menunjukkan kinerja moderat tetapi menunjukkan ruang untuk perbaikan dibandingkan dengan K-NN. Tabel 4.3 dan 4.4 menunjukkan performa Naive-Bayes pada data latih dan validasi.

Tabel 4.3 Performa algoritma Naive-Bayes pada data latih

Label	**Precision**	**Recall**	**F1-score**	**Support**
0	0.93	0.93	0.93	358
1	0.78	0.84	0.81	356
2	0.81	0.71	0.76	345
3	0.89	0.92	0.90	341
Accuracy	0.85			1400
Macro Avg	0.85	0.85	0.85	1400
Weighted Avg	0.85	0.85	0.85	1400

Tabel 4.4 Performa algoritma Naive-Bayes pada data validasi

Label	**Precision**	**Recall**	**F1-score**	**Support**
0	0.88	0.92	0.90	142
1	0.76	0.77	0.77	144
2	0.81	0.78	0.79	155
3	0.94	0.91	0.92	159
Accuracy	0.85			600
Macro Avg	0.85	0.85	0.85	600
Weighted Avg	0.85	0.85	0.85	600

Setelah melakukan *hyperparameter tuning* untuk model K-NN, ditemukan bahwa konfigurasi tertentu meningkatkan kinerja model. *Hyperparameter* yang optimal terlampir pada Tabel 4.5.

Tabel 4.5 *Hyperparameter* K-NN

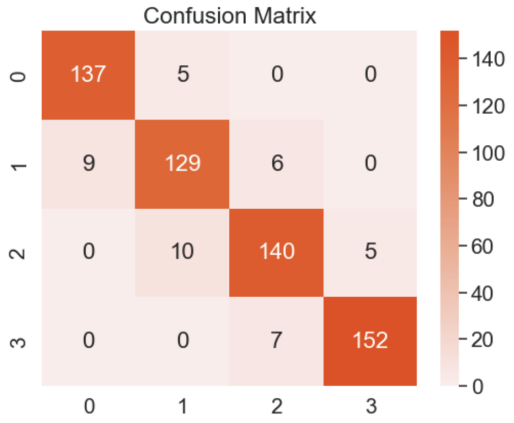
Parameter	**Nilai**
`k`	1
`metric`	minkowski
`p`	2
`weight`	uniform

Hyperparameter ini diidentifikasi sebagai yang paling baik dalam mencapai akurasi tertinggi dan kinerja yang seimbang di seluruh kelas. Pilihan k = 1 mengimplikasikan bahwa model ini bergantung pada tetangga terdekat untuk klasifikasi, sehingga sangat sensitif terhadap variasi lokal dalam data. Metrik Minkowski dengan p=2 sesuai dengan jarak Euclidean standar, yang sesuai untuk fitur kontinu. Selain itu, dengan menggunakan bobot yang seragam berarti semua tetangga berkontribusi secara merata dalam proses pengambilan keputusan, memberikan pengaruh yang adil dan tidak bias.

Secara singkat, model K-NN mengungguli Naive-Bayes dalam hal akurasi dan kinerja yang seimbang di seluruh kelas, menjadikannya pilihan yang lebih cocok untuk dataset ini. Temuan ini menekankan pentingnya memilih algoritma yang sesuai dengan karakteristik data, dan dalam hal ini, sifat K-NN yang sensitif secara lokal terbukti bermanfaat untuk mencapai hasil yang lebih baik.

4.2. *Confusion Matrix*

Confusion matrix adalah alat fundamental dalam evaluasi model klasifikasi, yang memberikan gambaran menyeluruh mengenai kinerja mereka. Matriks ini merupakan matriks persegi yang mendeskripsikan jumlah prediksi true positive (TP), true negative (TN), false positive (FP), dan false negative (FN). Setiap baris dari matriks mewakili kelas yang sebenarnya, sementara setiap kolom mewakili kelas yang diprediksi. Elemen diagonal (TP dan TN) menunjukkan prediksi yang benar, sedangkan elemen di luar diagonal (FP dan FN) menunjukkan contoh misklasifikasi. *Confusion matrix* berfungsi sebagai dasar untuk mendapatkan berbagai metrik kinerja seperti presisi, recall, akurasi, dan skor F1. Gambar 4.1 dan 4.2 menunjukkan performa model K-NN dan Naive-Bayes pada data validasi.



Gambar 4.1 Confusion Matrix K-NN

(Sumber: Dokumentasi Penulis)

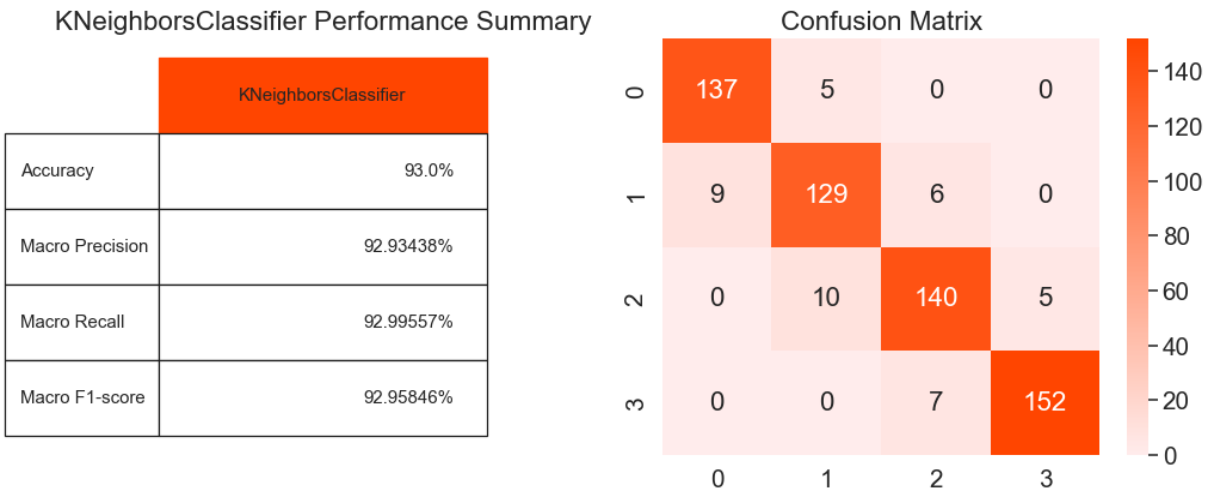
Gambar 4.2 Confusion Matrix Naive-Bayes

(Sumber: Dokumentasi Penulis)

Confusion matrix model K-NN menunjukkan performa yang cukup baik, dengan misklasifikasi terbanyak diakibatkan tertukarnya prediksi dan aktual dari kelas 1 dan 2. Pada sisi lain, *confusion matrix* model Naive-Bayes juga menghasilkan pola misklasifikasi yang mirip meskipun jumlahnya yang lebih banyak.

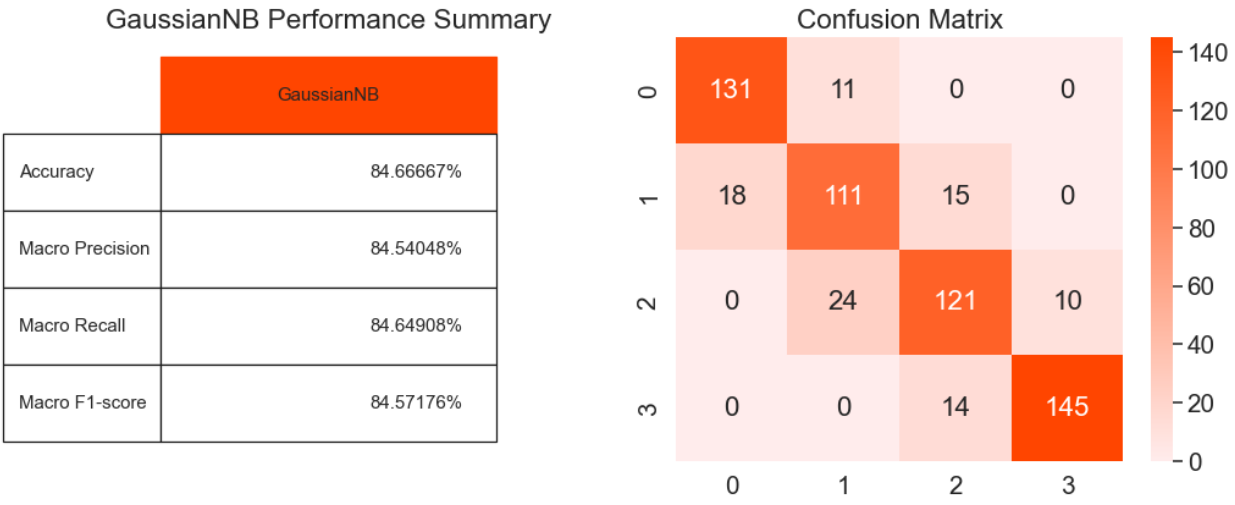
4.3. Perbandingan dengan Pustaka Scikit Learn

Implementasi model K-NN dan Naive-Bayes memiliki hasil yang sama persis untuk seluruh metrik dengan yang diaplikasikan Pustaka Scikit Learn. Hal ini disebabkan sifat algoritma dari K-NN dan Naive Bayes yang simpel dan tidak mengandung elemen *stochasticity* atau *randomness*. Gambar 4.3 dan 4.4 menunjukkan performa model K-NN dan Naive Bayes dari pustaka Scikit Learn.



Gambar 4.3 Performa model K-NN Scikit Learn pada data validasi

(Sumber: Dokumentasi Penulis)



Gambar 4.4 Performa model Naive Bayes Scikit Learn pada data validasi

(Sumber: Dokumentasi Penulis)

LAMPIRAN

Lampiran 1 Pranala GitHub *Repository*

https://github.com/AlifioDitya/KNN-NaiveBayes-AI

DAFTAR PUSTAKA

Awan, Abid Ali. (2023). *Naive Bayes Classification Tutorial using Scikit-learn*. https://www.datacamp.com/tutorial/naive-bayes-scikit-learn. Diakses pada 29 November 2023.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

Kontributor Data to Viz. *Density*. https://www.data-to-viz.com/graph/density.html. Diakses pada 8 November 2023.

Kontributor Open Genus. (2023). *Gaussian Naive Bayes*. https://iq.opengenus.org/gaussian-naive-bayes/. Diakses pada 29 November 2023.

Turner, Shaun. (2022). *What Is Kurtosis? Definition, Examples & Formula*. https://www.scribbr.com/statistics/kurtosis/. Diakses pada 8 November 2023.

Walpole, dkk. (2012). *Probability & Statistics for Engineer & Scientist 9th ed*. Pearson Education, Inc: United States of America.