IF3170 - Inteligensi Buatan
Implementasi Algoritma KNN dan Naive Bayes
Tugas Besar 2

Tanggal Pengumpulan: Rabu, 29 November 2023

Nama	NIM
Irsyad Nurwidianto Basuki	13521072
William Nixon	13521123
Nicholas Liem	13521135
I Putu Bakta Hari Sudewa	13521150

Institut Teknologi Bandung
Sekolah Teknik Elektro dan Informatika
Tahun Ajaran 2023/2024

Daftar Isi

Daftar Gambar

Daftar Tabel

1 Implementasi Algoritma

1.1 Implementasi Algoritma KNN

1.1.1 Deskripsi Singkat Algoritma KNN

Algoritma KNN atau yang biasa dikenal sebagai *K-Nearest Neighbor* adalah salah satu metode pembelajaran mesin yang bersifat *supervised learning*, juga *instance-based classifier* di mana algoritma ini menyimpan semua *training data*, dan juga algritma yang *lazy learner*. Algoritma ini biasanya digunakan untuk klasifikasi dan regresi. Algoritma KNN pada dasarnya akan memprediksi data *unseen* dengan mencari yang mirip dengan data tersebut dari data latih.

1.1.2 Bagaimana Proses Klasifikasi Algoritma KNN Bekerja

Iterasi dari algoritma KNN adalah sebagai berikut.

1. Lakukan perhitungan jarak antara *unseen data* dengan semua jarak pada data latih. Terdapat berbagai macam fungsi jarak yang bisa dipilih biasanya dapat menggunakan fungsi jarak Euclidean, Manhattan, Minkowski, atau Chebyshev.

2. Pilih k buah instans yang paling mirip.

3. Pilih *majority class* dari k buah instans yang dipilih menjadi hasil klasisifikasi.

Berikut adalah gambaran sederhana bagaimana Algoritma KNN bekerja dengan parameter $k = 3$ dan fungsi jarak yang dipakai adalah jarak Euclidean. Objektif atau fungsi target yang diinginkan adalah untuk menentukan kualitas roti berdasarkan fitur Warna, Ukuran, Tekstur, dan Berat.

Tabel 1: Tabel Data Latih KNN pada Data Roti

Sample	**Warna**	**Ukuran**	**Tekstur**	**Berat**	**Kualitas**
1	7	5	6	250	Baik
2	4	6	7	300	Buruk
3	5	7	4	220	Baik
4	8	4	5	260	Baik
5	6	3	8	280	Buruk

Tabel 2: Data Baru untuk Diklasifikasi

Warna	**Ukuran**	**Tekstur**	**Berat**	**Kualitas Prediksi**
5	5	5	240	?

Tabel 3: Jarak Data Baru terhadap Data Latih pada Data Roti

Sample	**Warna**	**Ukuran**	**Tekstur**	**Berat**	**Kualitas**	**Jarak ke Data Baru**
1	7	5	6	250	Baik	10.25
2	4	6	7	300	Buruk	60.05
3	5	7	4	220	Baik	20.12
4	8	4	5	260	Baik	20.25
5	6	3	8	280	Buruk	40.17

Tabel 4: Klasifikasi Data Baru dengan KNN

Tetangga Terdekat (Sample #)	**Kualitas**
1	Baik
3	Baik
4	Baik
Mayoritas Kualitas: Baik	

1.1.3 Kelebihan dan Kekurangan KNN

Kelebihan Algoritma KNN adalah sebagai berikut.

1. Aproksimasi yang dilakukan cenderung sedikit tidak kompleks untuk mendapatkan fungsi target.

Kekurangan Algoritma KNN adalah sebagai berikut.

1. Mahalnya klasifikasi data baru sebab harus melakukan perhitungan jarak dengan setiap data latih. Jika data latih besar, maka perhitungan jarak juga semakin banyak.

2. Algoritma ini mengkonsiderasi semua fitur (jika dibuat tanpa batasan fitur) padahal mungkin sebenarnya fungsi target hanya membutuhkan beberapa fitur saja.

1.2 Implementasi Algoritma Naive Bayes

1.2.1 Deskripsi Singkat Algoritma Naive Bayes

Algoritma Naive Bayes adalah metode pembelajaran mesin yang juga *supervised* dan menggunakan probabilitas sebagai sarana utama dalam menentukan pilihan dan biasanya digunakan untuk klasifikasi juga. Algoritma ini dibuat berdasarkan Teorema Bayes (*Conditional Probability*), walaupun algoritma ini sederhana tetapi cukup kuat dan efektif dalam memberikan klasifikasi berbasis probabilitas. Namun, pada algoritma ini, diasumsikan bahwa semua fitur yang digunakan dalam klasifikasi adalah independen satu sama lain yang bisa dampak baik maupun buruk tergantung pada jenis fitur dan data yang diberikan. Hal ini menyederhanakan perhitungan tetapi pada kenyataannya biasanya jadi jauh dengan hasil yang diinginkan.

1.2.2 Bagaimana Algoritma Naive Bayes Bekerja

Iterasi dari algoritma Naive Bayes adalah sebagai berikut.

1. Lakukan perhitungan probabilitas terjadinya setiap kelas $P(v_j)$.
2. Lakukan perhitungan probabilitas dari atribut *unseen data* untuk setiap kelas $\prod_i P(a_i|v_j)$
3. Kalikan setiap hasil dari probabilitas atribut (untuk semua atributnya) $P(v_j|a_1, a_2, \ldots, a_n) = P(v_j) \cdot \prod_i P(a_i|v_j)$: Produk dari semua probabilitas kondisional
4. Ambil nilai terbesar dari nilai probabilitas yang dihitung pada tahap sebelumnya
   $\max(P(v_j|a_1, a_2, \ldots, a_n), P(v_k|a_1, a_2, \ldots, a_n), \ldots)$

Keterangan tambahan untuk bagian sebelumnya.

1. $P(a_i|v_j)$: Probabilitas kondisional di mana a_i adalah sebuah fitur spesifik dari data yang belum dilihat (*unseen* data) dan v_j mewakili kelas atau nilai target tertentu. Kesimpulannya adalah probabilitas atribut a_i terjadi dengan asumsi bahwa kita tahu data tersebut adalah termasuk kelas v_j.
2. $\prod_i P(a_i|v_j)$: Produk atau perkalian dari semua probabilitas kondisional untuk setiap atribut a_i jika diberikan kelas v_j.
3. $P(v_j|a_1, a_2, \ldots, a_n)$: Probabilitas posterior atau probabilitas kelas v_j mengingat semua atribut $a_1, a_2, \ldots, a_n$ dari data yang belum terlihat. Hasilnya adalah perkalian antara probabilitas prior $P(v_j)$ dengan produk dari semua probabilitas kondisional atribut.
4. $\max(P(v_j|a_1, a_2, \ldots, a_n), P(v_k|a_1, a_2, \ldots, a_n), \ldots)$: Langkah pemilihan di mana algoritma akan memilih probabilitas posterior tertinggi sebagai hasil klasifikasi, jadi nanti akan dipilih probabilitas mana yang memiliki tertinggi maka kita akan ambil nilai v_k yakni sebuah kelas dengan probabilitas tertinggi diberikan atribut $a_1, a_2, \ldots, a_n$.

Berikut adalah gambaran sederhana bagaimana Algoritma Naive Bayes bekerja. Objektif atau fungsi target yang diinginkan adalah untuk menentukan kualitas roti berdasarkan fitur Warna, Ukuran, Tekstur, dan Berat.

Tabel 5: Tabel Data Latih Naive Bayes pada Data Roti

Sample	**Warna**	**Ukuran**	**Tekstur**	**Berat**	**Kualitas**
1	Cerah	Kecil	Kasar	Ringan	Baik
2	Gelap	Kecil	Lembut	Berat	Buruk
3	Cerah	Kecil	Kasar	Ringan	Baik
4	Gelap	Kecil	Kasar	Ringan	Baik
5	Cerah	Besar	Lembut	Berat	Buruk
6	Cerah	Kecil	Kasar	Ringan	Baik
7	Gelap	Besar	Lembut	Berat	Buruk

Dari Tabel 5, kita dapat melihat ada 4 atribut misalkan kita sebut mereka sebagai berikut.

1. a_1 adalah warna.
2. a_2 adalah ukuran.
3. a_3 adalah tekstur.
4. a_4 adalah berat.
5. v_i adalah kualitas baik.
6. v_j adalah kualitas buruk.

Berikut adalah data baru yang akan dicari kelas prediksinya apa.

Tabel 6: Data Baru untuk Diklasifikasi

Warna	**Ukuran**	**Tekstur**	**Berat**	**Kualitas Prediksi**
Cerah	Kecil	Lembut	Berat	?

Tabel 7: Probabilitas Kondisional untuk Naive Bayes

Atribut	Kondisi **A**	Kondisi **B**
P(Warna — Baik)	0.75	0.25
P(Warna — Buruk)	0.333	0.667
P(Tekstur — Baik)	1.0	-
P(Tekstur — Buruk)	-	1.0
P(Ukuran — Baik)	1.0	-
P(Ukuran — Buruk)	0.333	0.667
P(Berat — Baik)	1.0	-
P(Berat — Buruk)	-	1.0

Keterangan:

1. A: (Cerah/Kasar/Kecil/Ringan)
2. B: (Gelap/Lembut/Besar/Berat)

Data baru yang belum dilihat adalah sebagai berikut.

1. a_1 : Cerah
2. a_2 : Kecil
3. a_3 : Lembut
4. a_4 : Berat
5. $P(v_i) = 4/7$
6. $P(v_j) = 3/7$

Probabilitas totalnya adalah sebagai berikut (dengan asumsi nilai - digantikan dengan 1.0).

1. $P(v_i|a_1 = Cerah, a_2 = Kecil, a_3 = Lembut, a_4 = Berat) =$
   $P(v_i) \cdot P(a_1 = Cerah|v_i) \cdot P(a_2 = Kecil|v_i) \cdot P(a_3 = Lembut|v_i) \cdot P(a_4 = Berat|v_i) = \frac{4}{7} \cdot 0.75 \cdot 1.0 \cdot 1.0 \cdot 1.0 = 0.428$

2. $P(v_j|a_1 = Cerah, a_2 = Kecil, a_3 = Lembut, a_4 = Berat) =$
   $P(v_j) \cdot P(a_1 = Cerah|v_j) \cdot P(a_2 = Kecil|v_j) \cdot P(a_3 = Lembut|v_j) \cdot P(a_4 = Berat|v_j) = \frac{3}{7} \cdot 0.333 \cdot 0.333 \cdot 1.0 \cdot 1.0 = 0.047$

Maka hasil klasifikasinya adalah $\max(0.428, 0.047)$ yakni probabilitas dengan fitur v_i atau Baik.

1.2.3 Gaussian Naive Bayes Untuk Fitur Numerik

Ketika berhubungan dengan data numerik, pendekatan kategorikal yang digunakan pada bagian sebelumnya dapat menjadi tidak optimal. Hal ini karena frekuensi peluang menemukan nilai suatu a_i pada data kontinu sangatlah kecil, menyebabkan peluang terjadinya fitur tersebut mendekati 0. Gaussian Naive Bayes adalah salah satu jenis model Naive Bayes yang dapat digunakan untuk klasifikasi data dengan fitur numerik. Pada model ini, fitur diasumsikan memiliki distribusi Gaussian (normal). Pendekatannya serupa dengan bagian 1.2.2, akan tetapi dibutuhkan perkiraan rata-rata dan deviasi standar untuk mengestimasi probabilitas $P(a_i|V_j)$ di setiap kelas dengan menggunakan probabilitas Gaussian.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

Algoritma ini memodifikasi algoritma Naive Bayes Kategorical (1.2.2) pada langkah ke 3, yaitu pada perhitungan probabilitas terjadinya unseen data untuk setiap kelas. Algoritma hasil modifikasinya ialah sebagai berikut:

1. Hitung mean dari setiap fitur a_i untuk kelas v_j tertentu $\mu_{i,j}$.
2. Hitung variansi dari setiap fitur a_i untuk setiap kelas v_j tertentu $\sigma^2_{i,j}$.
3. Probabilitas $P(a_i|v_j)$ diperoleh dengan menggunakan fungsi probabilitas Gaussian $f(x_i|v_j) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu_{i,j}}{\sigma_{i,j}}\right)^2\right)$.
4. Kalikan setiap hasil dari probabilitas atribut (untuk semua atributnya) $P(v_j|a_1, a_2, \ldots, a_n) = P(v_j) \cdot \prod_i P(a_i|v_j)$: Produk dari semua probabilitas kondisional
5. Ambil nilai terbesar dari nilai probabilitas yang dihitung pada tahap sebelumnya
   $\max(P(v_j|a_1, a_2, \ldots, a_n), P(v_k|a_1, a_2, \ldots, a_n), \ldots)$

1.2.4 Kelebihan dan Kekurangan Naive Bayes

Kelebihan Naive Bayes

1. Efisien secara Komputasional: Naive Bayes cenderung lebih efisien secara komputasional dibandingkan dengan beberapa algoritma klasifikasi lainnya, terutama ketika datanya cukup besar. Hal ini karena hanya probabilitas saja yang disimpan.
2. Model Sederhana: Algoritma Naive Bayes relatif mudah diimplementasikan dan sederhana secara konsep. Ini membuatnya menjadi pilihan yang baik untuk tugas klasifikasi sederhana. Meskipun mengasumsikan independensi fitur, seringkali masih dapat memberikan kinerja yang baik.

Kekurangan Naive Bayes

1. Asumsi Independensi yang Naif: Asumsi bahwa setiap fitur bersifat independen, yang mendasari "naive" dalam Naive Bayes, tidak selalu berlaku di dunia nyata. Ini bisa menyebabkan kinerja yang buruk jika ada dependensi yang signifikan antara fitur.
2. Sensitif terhadap Perubahan Skala: Naive Bayes dapat menjadi sensitif terhadap perubahan skala karena mengandalkan perhitungan probabilitas. Oleh karena itu, normalisasi atau standarisasi fitur mungkin diperlukan.
3. Kurang baik pada data dengan berbagai jenis fitur: Naive Bayes memerlukan banyak asumsi distribusi maupun jenis fitur, misalnya kategorikal maupun numerik. Pada data dengan banyak jenis fitur dan beragam distribusi, model ini mungkin kurang baik.

1.2.5 Catatan

Meninjau bahwa pada data yang diberikan, fitur-fitur yang berkorelasi kuat dengan $price_range$ merupakan kategori numerik $['battery\_power','px\_height','px\_width','ram']$, maka pendekatan Gaussian Naive Bayes inilah yang diimplementasikan kelompok kami. Fitur kategorikal tidak dilibatkan karena tidak ditemukan korelasi yang kuat dengan variabel target, dan juga karena keterbatasan algoritma Gaussian Naive Bayes untuk data kategori. Agar memenuhi asumsi distribusi Gaussian pada data, kami menggunakan Standard Scaler pada fitur terlebih dahulu. Scaler ini difit pada data latih, dan digunakan untuk menscale data validasi.

2 Perbandingan Hasil Prediksi Algoritma

Pada bagian ini, akan dibahas mengenai perbandingan dari hasil prediksi algoritma yang diimplementasi secara manual dengan algoritma dari pustaka (*library*) yang ada di Python. Hal yang akan dibandingkan mencakup *precision*, *recall*, dan *accuration*.

- Precision, mengukur sejauh mana hasil positif dari model benar-benar relevan. Secara matematis, precision dapat dihitung dengan rumus:

$$Precision = \frac{TruePositives}{(TruePositives + FalsePositives)}$$

  Ketika membandingkan implementasi manual dan pustaka, perhatikan apakah kedua pendekatan memberikan nilai precision yang serupa.

- Recall, mengukur sejauh mana model dapat mengidentifikasi semua kasus positif yang sebenarnya. Rumusnya adalah:

$$Recall = \frac{TruePositives}{(TruePositives + FalseNegatives)}$$

  Sama seperti precision, perbandingan recall antara implementasi manual dan pustaka dapat memberikan wawasan tentang kinerja keduanya.

- Accuracy, metrik umum yang mengukur sejauh mana model dapat memprediksi dengan benar keseluruhan kelas. Rumusnya adalah:

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalInstances}$$

  Perbandingan akurasi antara implementasi manual dan pustaka akan memberikan gambaran tentang sejauh mana keduanya setuju dalam memprediksi kelas.

Evaluasi Tambahan selain ketiga metrik di atas, yaitu F1-score atau area di bawah kurva ROC (Receiver Operating Characteristic) juga digunakan untuk memberikan gambaran komprehensif tentang kinerja model.

2.1 Perbandingan Algoritma KNN

Perbandingan hasil prediksi algoritma KNN dilakukan untuk algoritma yang diimplementasi secara manual dan yang diimplementasi menggunakan pustaka (*library*). Untuk dapat membandingkan hasil prediksi kedua implementasi tersebut, digunakan fungsi "classification_report" dari pustaka sklearn. Berikut merupakan hasil dari penggunaan "classification_report" pada kedua implementasi:

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.98	0.96	142
1	0.91	0.87	0.89	144
2	0.89	0.91	0.90	155
3	0.97	0.94	0.96	159
accuracy			0.93	600
macro avg	0.92	0.93	0.92	600
weighted avg	0.93	0.93	0.92	600

Gambar 1: KNN Implementasi Secara Manual

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.98	0.96	142
1	0.91	0.87	0.89	144
2	0.89	0.91	0.90	155
3	0.97	0.94	0.96	159
accuracy			0.93	600
macro avg	0.92	0.93	0.92	600
weighted avg	0.93	0.93	0.92	600

Gambar 2: KNN Implementasi Dengan Pustaka

Hasil dari penggunaan "classification_report" pada kedua implementasi menunjukkan bahwa algoritma implementasi KNN yang dibuat secara manual sudah menghasilkan hasil yang sama dengan implementasi KNN dengan menggunakan pustaka yang disediakan sklearn. Hal tersebut dapat dilihat pada gambar atau file .ipynb yang menunjukkan data *precision*, *recall*, dan *accuracy* menghasilkan hasil yang sama, pada implementasi KNN manual maupun implementasi KNN dengan pustaka.

2.2 Perbandingan Algoritma Naive Bayes

Perbandingan hasil prediksi algoritma Naive Bayes dilakukan untuk algoritma yang diimplementasi secara manual dan yang diimplementasi menggunakan pustaka (*library*). Untuk dapat membandingkan hasil prediksi kedua implementasi tersebut, digunakan fungsi "classification_report" dari pustaka sklearn. Berikut merupakan hasil dari penggunaan "classification_report" pada kedua implementasi:

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.88	0.88	143
1	0.65	0.67	0.66	140
2	0.72	0.68	0.70	163
3	0.88	0.91	0.89	154
accuracy			0.79	600
macro avg	0.78	0.79	0.78	600
weighted avg	0.78	0.79	0.78	600

Gambar 3: Hasil Algoritma Naive Bayes dengan Implementasi Manual

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

           0       0.88      0.89      0.88       142
           1       0.67      0.65      0.66       144
           2       0.68      0.72      0.70       155
           3       0.91      0.88      0.89       159

    accuracy                           0.79       600
   macro avg       0.79      0.78      0.78       600
weighted avg       0.79      0.79      0.79       600
```

Gambar 4: Hasil Algoritma Naive Bayes dengan Pustaka

Dari Gambar 3 dan 4 terdapat beberapa perbedaan pada nilai *precision*, *recall*, dan *support* yang diperoleh. Namun, dalam kasus ini, analisis akan difokuskan kepada perbandingan nilai *precision*, *recall*, dan *accuracy* dari kedua hasil. Untuk algoritma Naive Bayes yang diimplementasikan secara manual (Gambar 3), nilai *precision* untuk *cluster* 0 (biaya rendah) dan *cluster* 2 (biaya tinggi) lebih tinggi dibandingkan nilai *precision* algoritma Naive Bayes yang diimplementasikan dengan pustaka *sklearn*. Sedangkan untuk *cluster* lainnya, yaitu *cluster* 1 dan 3, algoritma Naive Bayes yang diimplementasikan dengan pustaka memiliki nilai *precision* yang lebih tinggi dibandingkan dengan algoritma Naive Bayes yang diimplementasikan secara manual. Berikutnya, terkait *recall*, algoritma Naive Bayes yang diimplementasikan secara manual (Gambar 3) memiliki nilai *recall* yang lebih tinggi hanya pada cluster 2 (biaya tinggi) saja. Selanjutnya, terkait *accuracy*, baik algoritma Naive Bayes yang diimplementasikan secara manual maupun dengan pustaka, memiliki nilai *accuracy* yang sama, yaitu 0.79.

Dari kedua hasil tersebut dapat disimpulkan model yang dihasilkan dengan menggunakan algoritma Naive Bayes secara manual sudah ”cukup baik” karena perbedaan hasil yang diperoleh dengan algoritma Naive Bayes yang menggunakan pustaka tidak jauh berbeda, bahkan terdapat beberapa komponen yang hasilnya lebih baik. Selain itu, nilai *accuracy* yang sama antara kedua pendekatan (manual dan pustaka) juga memperkuat kesimpulan tersebut.

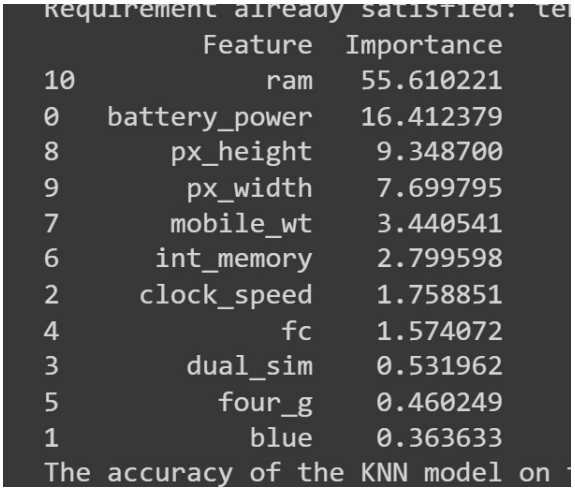
3 Pemrosesan Kaggle Submission

Pada bagian ini, akan dibahas pemrosesan data tambahan pada algoritma KNN. Algoritma KNN dipilih karena dirasa sebagai model yang lebih tepat pada dataset ini. Hal ini disebabkan beberapa hal, yaitu pertama, karena model ini memperoleh akurasi yang lebih baik ($> 0.9\%$) pada data validasi tanpa perlu pemrosesan lanjut, tidak seperti algoritma Naive Bayes hasil implementasi yang relatif lebih buruk (80%). Kedua, karena pada algoritma KNN hasil implementasi memiliki lebih banyak parameter seperti K, Distance Metric, Weighing Features, dan seterusnya.

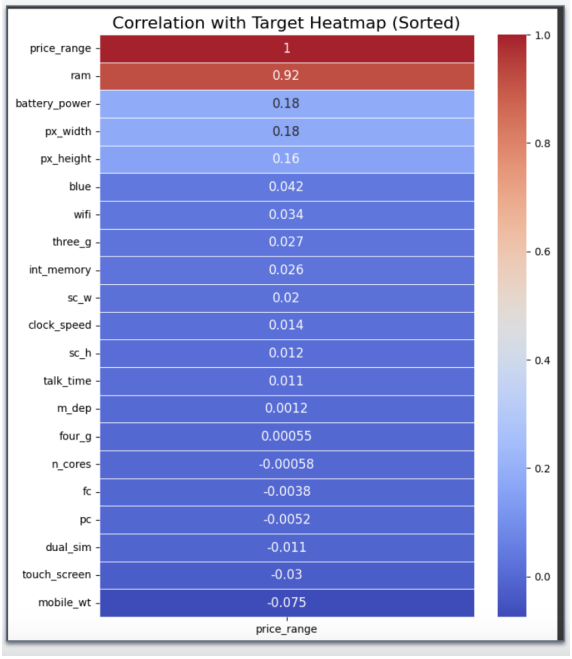
3.1 Data Preprocessing

3.1.1 Feature Selection

Feature selection dipilih berdasarkan kolom-kolom yang dirasa memiliki kontribusi yang cukup besar terhadap prediksi `price_range`. Kolom tersebut ialah sebagai berikut: `selected_features = ['battery_power', 'area', 'ram', 'px_width', 'px_height', 'mobile_wt', 'int_memory']`. Adapun referensi pemilihan kolom tersebut yaitu berdasarkan hasil korelasi, dan hasil `'feature_importance'` dari model CatBoost.



Gambar 5: Feature Importance CatBoost



Gambar 6: Heatmap Korelasi

3.1.2 Feature Engineering

Feature Engineering yang dilakukan pada data ini melibatkan beberapa langkah, seperti penambahan fitur baru ("area") dan normalisasi fitur-fitur tertentu menggunakan Min-Max Scaling. Berikut adalah penjelasan langkah-langkah preprocessing yang dilakukan:

1. Penambahan fitur baru area: Dalam fungsi add_area, sebuah fitur baru bernama "area" ditambahkan ke setiap data frame (df). Fitur ini dihitung dengan mengalikan lebar piksel (px_width) dengan tinggi piksel (px_height). Tujuan penambahan fitur ini mungkin untuk menggambarkan luas tampilan layar atau aspek lain yang berkaitan dengan kedua fitur tersebut.

2. Normalisasi fitur-fitur terpilih: Dilakukan normalisasi terhadap fitur-fitur yang terpilih menggunakan Min-Max Scaling. Jika parameter train bernilai True, maka normalisasi dilakukan dengan membuat dan meng-fit ulang sebuah objek MinMaxScaler. Jika parameter train adalah False, maka normalisasi dilakukan menggunakan objek MinMaxScaler yang telah di-fit sebelumnya.

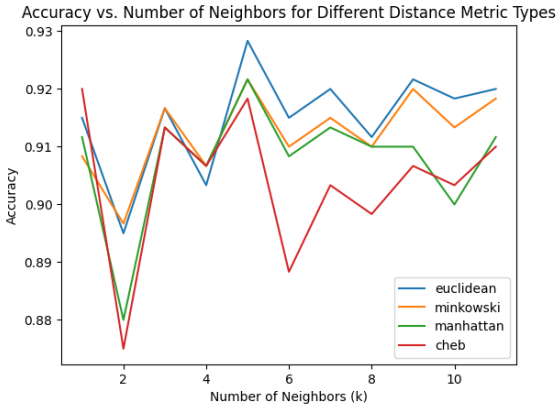
3. Weighting: Akan dilakukan feature weighting pada setiap kolom yang terpilih. Nilai weighting ini ditentukan secara eksprimental secara arbitrer. Ini dimulai dengan menentukan 3 variabel terlebih dahulu yaitu "ram", "area", dan "battery_power", lalu menjalankan for loop dari range 1-10 dengen inkremen 1 dan mengambil hasil terbaik. Setelah itu, nilai 3 variabel awal tersebut di tetapkan, dan mulai ditambahkan kolom-kolom lain dengan iterasi for loop 0.1 - 2 dengan inkremen 0.1. Adapun hasil weighting yang berhasil memberikan

hasil yang diinginkan yaitu

$$\begin{aligned}\text{weights} = \{&\text{"ram"}: 5,\\ &\text{"area"}: 4,\\ &\text{"battery\_power"}: 1,\\ &\text{'px\_width'}: 0.9,\\ &\text{'px\_height'}: 1.1,\\ &\text{'mobile\_wt'}: 0.1,\\ &\text{'int\_memory'}: 0.1\}\end{aligned}$$

3.2 Hyperparameter Tuning (KNN Model)

Pada bagian ini, akan ditentukan konfigurasi untuk nilai K neighbor dan juga distance_metric yang akan dipilih. Adapun hasil eksperimen dapat dilihat pada gambar di bawah ini.



Gambar 7: Hasil Eksperimen Tuning untuk Berbagai Metrik Distance

Dari gambar ini, dapat dilihat bahwa distance metric Euclidean memiliki performa yang relatif lebih konsisten dibandingkan distance metric lain. Hal ini mungkin dikarenakan pembobotan weighting ditentukan dengan menetapkan distance sebagai "Euclidean", sehingga hasil weighting tersebut tentunya optimal untuk distance metric "Euclidean".

Selain itu, nilai neighbor terbaik nampaknya sulit ditentukan, mengingat tidak adanya "shoulder" pada grafik ini. Oleh karena itu, kami akan mencoba beberapa nilai K yang menjadi 'lokal optima', yaitu pada saat data bernilai 1, 3, dan 5.

3.2.1 Submisi Kaggle

Kami melakukan beberapa submisi Kaggle untuk melihat parameter dengan nilai terbaik pada data tes. Nilai weights tetap, dan kami menvariasikan distance metric dan banyak neighbors yang digunakan. Adapun submisi terbaik kami saat ini diperoleh dengan nilai akurasi 0.974, dengan nilai neighbor 3 dan distance metric Minkowski. Penulis memohon maaf karena rekapitulasi yang lengkap tidak dapat disajikan di laporan ini, mengingat keterbatasan waktu dan tenaga penulis, .

4 Kontribusi Anggota Kelompok

NIM	Nama	Pengerjaan
13521072	Irsyad Nurwidianto Basuki	KNN SkLearn, NB SkLearn
13521123	William Nixon	Naive Bayes Scratch, Naive Bayes SkLearn, KNN with Weighting, Kaggle Processing
13521135	Nicholas Liem	KNN Scratch, Naive Bayes Scratch, Kaggle Processing
13521150	I Putu Bakta Hari Sudewa	KNN SkLearn, NB SkLearn

Repository: https://github.com/NicholasLiem/IF3150_M02_KNN_NB_Model

5 Daftar Pustaka

Exploratory Data Analysis. Year or Access Date. URL: https://en.wikipedia.org/wiki/Exploratory_data_analysis.

Indonesia, Data Folks. *Memahami Data dengan Exploratory Data Analysis*. Medium. Year or Access Date. URL: https://medium.com/data-folks-indonesia/memahami-data-dengan-exploratory-data-analysis-a53b230cce84.

Silva, Thiago. *Applying Heatmaps for Categorical Data Analysis*. Kaggle. Year or Access Date. URL: https://www.kaggle.com/code/tsilveira/applying-heatmaps-for-categorical-data-analysis.