IJAI (Indonesian Journal of Applied Informatics)
Vol. 9 No. 1 Tahun 2024 pISSN: 2548-3846, eISSN: 2598-5981

Klasifikasi Penyakit Anemia Menggunakan Algoritma *Navie Bayes*

Elda Putri Darmayanti[1]*, Ika Nur Fajri[1]
[1]Program Studi Sistem Informas, Ilmu Komputer, Universitas Amikom Yogyakarta, Indonesia
*Email: putrielda199@students.amikom.ac.id

Info Artikel

Kata Kunci :
anemia, algoritma klasifikasi, machine learning, naïve bayes, data mining, prediksi

Keywords :
anaemia, classification algorithm, machine learning, naïve bayes, data mining, prediction

Tanggal Artikel
Dikirim : 5 November 2024
Direvisi : 16 November 2024
Diterima : 17 November 2024

Abstrak

Anemia merupakan kondisi medis yang umum di mana darah seseorang kekurangan sel darah merah yang sehat atau hemoglobin. Hemoglobin adalah protein dalam sel darah merah yang berfungsi untuk mengangkut oksigen dari paru-paru ke seluruh tubuh ketika seseorang terkena anemia, mereka mungkin merasa lelah, lemah, dan sesak napas. Anemia dapat disebabkan oleh berbagai faktor, termasuk kekurangan zat besi, vitamin B12, atau folat; kehilangan darah; dan kerusakan sumsum tulang. Dalam upaya untuk meningkatkan diagnosis awal dan akurasi klasifikasi penyakit anemia, penelitian ini menerapkan algoritma Naïve Bayes. Dataset yang digunakan dalam penelitian ini adalah dataset penyakit anemia yang didapatkan dari *website* kaggle.com, yang mencakup atribut-atribut penting seperti Gender, Hemoglobin, MCH, MCHC, MCV, dan Result. Pemilihan Naïve Bayes sebagai salah satu algoritma yang diuji didasarkan pada keunggulannya dalam menangani data dengan atribut sederhana serta kemampuannya mengelola data yang mengandung ketidakpastian. Naïve Bayes dikenal sebagai algoritma yang efisien untuk pengolahan dataset berukuran besar dengan struktur data yang sederhana. Selain itu, algoritma ini sering menjadi pilihan pada tahap awal eksplorasi data karena kesederhanaan implementasi, kecepatan pemrosesan, dan kemampuannya menghasilkan hasil yang cukup akurat dalam berbagai kondisi. Meskipun Naïve Bayes mungkin tidak selalu lebih akurat daripada SVM atau *Decision Tree* dalam kasus kompleks, algoritma ini menawarkan solusi yang lebih cepat, ringan, dan mudah diimplementasikan, yang sangat relevan untuk aplikasi medis dengan sumber daya terbatas. Pemilihan Naïve Bayes dalam penelitian ini bertujuan untuk mengeksplorasi keseimbangan antara kecepatan, efisiensi, dan akurasi dalam klasifikasi penyakit anemia.

Abstract

Anaemia is a common medical condition where a person's blood lacks healthy red blood cells or haemoglobin. Haemoglobin is a protein in red blood cells that serves to transport oxygen from the lungs to the rest of the body. When a person is anaemic, they may feel tired, weak, and short of breath. Anaemia can be caused by various factors, including iron, vitamin B12, or folate deficiency; blood loss; and bone marrow damage. In an effort to improve the early diagnosis and classification accuracy of anaemia, this study applied the Naïve Bayes algorithm. The dataset used in this research is an anaemia disease dataset obtained from the website kaggle.com, which includes important attributes such as Gender, Haemoglobin, MCH, MCHC, MCV, and Result. The selection of Naïve Bayes as one of the tested algorithms is based on its superiority in handling data with simple attributes and its ability to manage data containing uncertainty. Naïve Bayes is known as an efficient algorithm for processing large datasets with simple data structures. Moreover, it is often the algorithm of choice in the early stages of data exploration due to its simplicity of implementation, processing speed, and ability to produce reasonably accurate results under various conditions. While Naïve Bayes may not always be more accurate than SVM or Decision Tree in complex cases, it does offer a bargain.

1. PENDAHULUAN

Anemia adalah kondisi medis yang ditandai dengan rendahnya kadar hemoglobin atau jumlah sel darah merah dalam tubuh, sehingga mengurangi kemampuan darah untuk mengangkut oksigen ke jaringan tubuh. Berdasarkan data terbaru dari Organisasi Kesehatan Dunia (WHO), prevalensi anemia secara global mengalami peningkatan, terutama di negara-negara berkembang, dengan sekitar 24,8% populasi dunia terdampak oleh kondisi ini (WHO, 2020). Penyebab anemia beragam, meliputi kekurangan zat besi, defisiensi vitamin, penyakit kronis, dan gangguan genetik (Smith & Zhang, 2021). Oleh karena itu, deteksi dini anemia sangat penting untuk mencegah komplikasi dan mendukung pengobatan yang efektif[1].

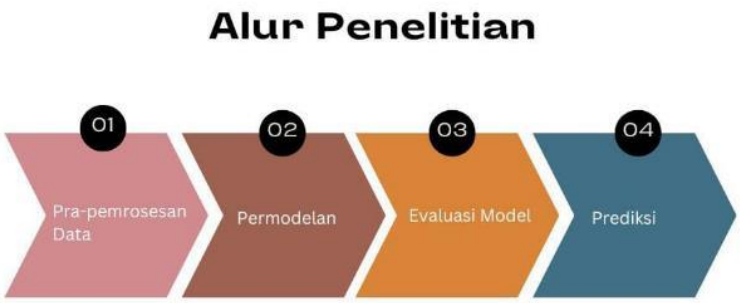
Penyakit anemia dapat berdampak signifikan pada kesehatan dan kualitas hidup seseorang, dengan berbagai dampak yang bergantung pada tingkat keparahan dan penyebab anemia. Salah satu dampak utama anemia adalah menurunnya kemampuan darah untuk mengangkut oksigen ke seluruh tubuh, yang dapat menyebabkan kelelahan berkepanjangan, kekurangan energi, dan penurunan daya tahan tubuh. Kekurangan oksigen ini juga dapat memengaruhi fungsi otak, mengakibatkan gangguan kognitif seperti kesulitan berkonsentrasi, kebingungan, dan penurunan daya ingat. Pada beberapa kasus, anemia dapat meningkatkan frekuensi dan intensitas pernapasan, terutama selama aktivitas fisik, karena tubuh berusaha meningkatkan pasokan oksigen, Dampak anemia bisa bervariasi, dan gejala yang dialami setiap individu mungkin berbeda[2]. Informasi mengenai penyakit anemia masih terbatas, sehingga banyak masyarakat yang belum sepenuhnya memahami kondisi ini. Umumnya, penderita anemia berkonsultasi langsung dengan dokter untuk mendapatkan diagnosis. Namun, peningkatan jumlah pasien dapat menyebabkan proses diagnosis menjadi tertunda. Oleh karena itu, diperlukan solusi yang memungkinkan diagnosis awal anemia secara mandiri sebelum pasien melakukan pemeriksaan lebih lanjut ke rumah sakit atau dokter[3].

Gejala anemia kini dapat diprediksi dengan bantuan teknologi yang mengumpulkan dan mengolah data historis menjadi informasi yang berguna. Klasifikasi risiko sangat penting untuk dilakukan guna memastikan penanganan yang tepat dan efektif, serta mencegah komplikasi lebih lanjut. Proses klasifikasi ini dapat dilakukan dengan memanfaatkan teknik "*data mining*" yang efektif dalam mengidentifikasi pola dan risiko berdasarkan data yang tersedia. *Data mining* adalah serangkaian proses untuk mengekstrak pola atau informasi yang berguna dari sejumlah besar data yang sebelumnya tidak terungkap secara manual, dengan tujuan untuk menghasilkan wawasan atau pengetahuan yang dapat digunakan untuk pengambilan keputusan. Ada beberapa metode algoritma klasifikasi yang umum digunakan dalam *data mining*, yaitu Naive Bayes, *Decision Tree*, *Support Vector Machine* (SVM), dan *k-Nearest Neighbors* (k-NN)[4]. Dalam penelitian ini digunakan metode Navie Bayes, Metode ini didasarkan pada Teorema Bayes, sebuah prinsip dalam statistik yang digunakan untuk menghitung probabilitas suatu kejadian berdasarkan probabilitas kejadian lain yang terkait. Metode Navie Bayes memiliki beberapa keunggulan, antara lain mudah diimplementasikan, cepat dalam menghasilkan prediksi, serta dapat diterapkan pada berbagai jenis data, termasuk data dengan jumlah fitur yang banyak[5].

Sebagai perbandingan, algoritma *Decision Tree* dipilih karena kemampuannya dalam menangani data yang kompleks, kemudahan dalam interpretasi model, serta kemampuannya untuk memberikan keputusan yang transparan melalui visualisasi pohon keputusan yang jelas. *Decision Tree* sering dianggap lebih unggul dalam menangani dataset dengan atribut yang saling berinteraksi atau memiliki hubungan yang kompleks. Berdasarkan hasil analisis dan pengujian, *Decision Tree* terbukti menunjukkan kinerja yang lebih baik dalam hal akurasi dibandingkan dengan algoritma Naïve Bayes[6][7]. Penelitian ini mengidentifikasi terkait kebutuhan akan algoritma yang sederhana namun tetap efektif dalam klasifikasi anemia, khususnya untuk data dengan atribut yang terbatas. Algoritma Naïve Bayes dipilih untuk mengeksplorasi sejauh mana pendekatan probabilistik ini dapat memberikan kinerja yang kompetitif dibandingkan dengan algoritma lain dalam konteks klasifikasi penyakit anemia. Klasifikasi penyakit anemia menggunakan algoritma Naive Bayes bertujuan untuk meningkatkan kualitas dan efektivitas dalam memprediksi kategori atau kelas dari data penderita anemia. Klasifikasi adalah proses mengelompokkan data berdasarkan karakteristik tertentu, seperti kategori atau kelas. [5]. Hasil penelitian menunjukan akurasi model sebesar 0,62 atau 62,45%, *Precision* dan *recall* positif (1,0) dan *negative* (0,0). Secara keseluruhan, model Naive Bayes menunjukkan kinerja yang cukup baik dalam memprediksi keberadaan atau tidaknya penyakit anemia. Penelitian ini diharapkan dapat membantu meningkatkan kualitas pelayanan kesehatan dan meningkatkan kesadaran masyarakat tentang pentingnya diagnosis dan pengobatan penyakit anemia yang tepat.

2. METODE PENELITIAN

Metode penelitian ini melibatkan beberapa tahapan diantaranya yaitu, Pra-pemrosesan data, Permodelan, Evaluasi Model, dan Prediksi. Penelitian ini dapat diliat pada Gambar 1. Setiap langkah dalam alur proses sangat penting untuk dilaksanakan. Penjelasan rinci mengenai masing-masing proses akan disampaikan dalam bab ini. Berikut adalah Gambar 1 yang menunjukkan alur metodologi penelitian.



Gambar 1. Alur Metodologi Penelitian

2.1 Pra-pemrosesan Data

Tahap awal dalam penelitian ini adalah pengumpulan data, yang menjadi dasar untuk analisis dan pengambilan hasil. Dalam penelitian ini, data yang digunakan berasal dataset yang diperoleh dari platform Kaggle, sebuah situs yang menyediakan berbagai dataset untuk keperluan pembelajaran dan analisis (https://www.kaggle.com/code/emreiekyurt/anemia-classification-with-eda-100-acc/input). Berisi 1536 data dengan 6 atribut dan 1 label kelas, berikut tahap-tahap pra-pemrosesan data :

1. Pemilihan fitur dan label

```
# Menentukan Variabel X (Fitur/Atribut) dan Variabel y (Kelas/Label)

X= data.iloc[:, :-1]
y= data.values[:, -1]

pd.DataFrame(y).head()
```

	0
0	0.0
1	0.0
2	1.0
3	0.0
4	0.0

Keterangan : Variabel X adalah fitur-fitur dari dataset data. Dalam konteks ini, menggunakan metode *.iloc[]* dari Pandas untuk memilih semua baris (:), tetapi mengambil semua kolom kecuali kolom terakhir (:-1). Ini berarti kita mengambil semua kolom kecuali kolom label, : Variabel y adalah target atau label dari dataset data. Di sini, menggunakan metode *.values* dari Pandas untuk mengonversi *DataFrame* menjadi *array NumPy*, kemudian memilih kolom terakhir (-1), yang merupakan kolom label, dan *pd.DataFrame(y).head():* Ini hanya digunakan untuk menampilkan lima baris pertama dari label y sebagai DataFrame baru untuk melihatnya dengan jelas.

2. *Train Test Split* / Pembagian dataset

```
[ ] # Memisahkan data menjadi data latih dan data uji
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Keterangan : *X_train, X_test, y_train, y_test:* Variabel ini akan menampung data latih dan data uji untuk fitur (X) dan label (y) secara terpisah, dan *train_test_split(X, y, test_size=0.2, random_state=42):* Fungsi ini membagi dataset menjadi data latih dan data uji. Argumen X adalah fitur-fitur dataset, y adalah labelnya. *test_size=0.2* menentukan bahwa 20% dari data akan digunakan sebagai data uji, sedangkan 80% sisanya akan digunakan sebagai data latih. *random_state=42* digunakan untuk membuat pembagian data yang konsisten ketika kode dieksekusi ulang.

2.2 Permodelan

Setelah menyelesaikan tahap awal dan berhasil memperoleh data dari *website* Kaggle, langkah selanjutnya adalah Permodelan. Pemodelan adalah proses penting dalam pembelajaran mesin dan analisis data yang melibatkan penggunaan algoritma untuk membuat prediksi atau mengidentifikasi pola dari data yang telah diproses. Proses pemodelan terdiri dari beberapa langkah kunci yang memastikan bahwa model yang dihasilkan memiliki akurasi tinggi dan dapat diandalkan untuk digunakan pada data baru.

Langkah pertama dalam permodelan adalah memilih algoritma atau jenis model yang sesuai dengan masalah yang dihadapi. Salah satu algoritma yang dapat dipertimbangkan adalah Naïve Bayes. Pemilihan model ini didasarkan pada sifat data dan tujuan prediksi yang ingin dicapai. Setelah memilih model, langkah berikutnya adalah melatih model menggunakan data pelatihan. Dalam proses ini, model mempelajari pola-pola dalam data dengan menyesuaikan parameter-parameter internalnya untuk meminimalkan kesalahan prediksi. Proses pelatihan ini melibatkan penggunaan metode optimasi seperti gradient descent untuk memperbaiki parameter-parameter tersebut sehingga model dapat membuat prediksi yang lebih akurat.

2.3 Evaluasi Model

Kinerja model dievaluasi menggunakan data pengujian yang terpisah. Evaluasi ini bertujuan untuk mengukur seberapa baik model dapat memprediksi data baru yang belum pernah dilihat sebelumnya. Metode evaluasi ini menggunakan berbagai metrik seperti akurasi, *presisi, recall, F1-score* untuk masalah klasifikasi, dan *mean squared error* (MSE) serta metrik regresi lainnya, tergantung pada jenis masalah yang sedang dihadapi. *Confusion Matrix* merupakan alat pengukur yang dapat digunakan untuk menghitung kinerja atau Tingkat kebenaran proses klasifikasi.

Tabel 1. Tabel *Classifier*

	Positif	**Negatif**
Positif	*TP*	*FP*
Negatif	*FN*	*TN*

Tabel 1 menunjukkan istilah-istilah yang digunakan untuk menganalisis performa sebuah *classifier*. Yang berisi yaitu, TP (*True Positive*), FN (*False Negatif*), FP (*False Positive*), dan TN (*True Negative*). Penilaian kinerja didasarkan pada perhitungan rata-rata beberapa metrik kinerja, seperti berikut:

1. Akurasi : Merupakan metrik evaluasi yang mengukur tingkat kedekatan antara nilai prediksi dengan nilai aktual. Akurasi dihitung berdasarkan proporsi jumlah data yang diklasifikasikan dengan benar terhadap total data. Dengan mengetahui jumlah data yang diklasifikasikan secara benar, kita dapat menentukan tingkat akurasi dari hasil prediksi model. Rumus akurasi adalah pada rumus 1.

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \; x \; 100 \qquad (1)$$

2. Presisi : Merupakan metrik evaluasi yang mengukur proporsi informasi relevan 14 yang berhasil diidentifikasi oleh sistem dibandingkan dengan total informasi yang diambil oleh sistem, baik yang relevan maupun tidak relevan. Presisi dihitung dengan membandingkan jumlah prediksi positif yang benar dengan total prediksi positif yang dihasilkan oleh model. Rumus presisi adalah pada rumus 2.

$$\text{Presisi} = \frac{TP}{TP+FP} \qquad (2)$$

3. *Recall* : Merupakan metrik evaluasi yang mengukur proporsi informasi relevan yang berhasil diidentifikasi oleh sistem dibandingkan dengan total informasi relevan yang sebenarnya ada dalam koleksi informasi. *Recall* dihitung dengan membandingkan jumlah prediksi positif yang benar dengan total jumlah data positif yang sebenarnya, termasuk yang tidak terambil oleh sistem. Rumus recall adalah pada rumus 3.

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

4. *F1-Score* : Merupakan sebuah metrik yang memberikan rata-rata harmonis antara presisi dan *recall*. Skor F1 yang tinggi menunjukkan bahwa model memiliki keseimbangan yang baik antara mengidentifikasi data positif dengan benar dan meminimalkan kesalahan prediksi. Rumus *F1-Score* adalah pada rumus 4.

$$F1\text{-}Score = 2\ x\ \frac{Presisi\ x\ Recall}{Presisi+Recall} \quad (4)$$

2.4 Prediksi

Tahap terakhir dalam penelitian ini adalah pengembangan antarmuka prediksi. *Deployment model* merupakan tahap akhir dan salah satu yang paling menantang dalam penerapan *Machine Learning*. *Deployment* adalah proses mengimplementasikan model pada lingkungan produksi agar dapat memberikan prediksi yang dapat diakses oleh sistem perangkat lunak. Tahap ini penting dilakukan untuk mendapatkan prediksi yang andal dan memaksimalkan nilai model *Machine Learning* yang telah dikembangkan. Tujuan dari *deployment model* ini adalah memastikan bahwa model yang telah dibuat dapat diakses dan digunakan oleh pengguna, sehingga mereka dapat memanfaatkan teknologi ini dalam kehidupan sehari-hari.

3. HASIL DAN PEMBAHASAN

Hasil dan pembahasan dalam penelitian ini, berdasarkan metode penelitian yang telah dijelaskan sebelumnya, menunjukkan bahwa klasifikasi risiko penyakit anemia menggunakan algoritma Navie Bayes menghasilkan beberapa temuan yang signifikan.

3.1 Pra-Pemrosesan Data

Tahap ini merupakan langkah awal dalam penelitian ini. Emreiekyurt adalah dataset penyakit anemia yang didapatkan dari *website* kaggle.com. Dataset yang digunakan dalam penelitian ini berisi 1536 data dengan 6 atribut dan 1 label kelas. Atribut yang digunakan meliputi Gender, Hemoglobin, MCH, MCHC, MCV, dan Result.

	Gender	Hemoglobin	MCH	MCHC	MCV	Result
0	1	14.9	22.7	29.1	83.7	0
1	0	15.9	25.4	28.3	72.0	0
2	0	9.0	21.5	29.6	71.2	1
3	0	14.9	16.0	31.4	87.5	0
4	1	14.7	22.0	28.2	99.5	0
...	...	...	...	...	...	...
1416	0	10.6	25.4	28.2	82.9	1
1417	1	12.1	28.3	30.4	86.9	1
1418	1	13.1	17.7	28.1	80.7	1
1419	0	14.3	16.2	29.5	95.2	0
1420	0	11.8	21.2	28.4	98.1	1

1421 rows × 6 columns

Gambar 2. Dataset penyakiit anemia

3.2 Permodelan

Setelah tahap pengumpulan data, langkah selanjutnya adalah Permodelan. Pembentukan model pada penelitian ini melibatkan beberapa tahap, termasuk tahap *Pre-processing* (Menentukan Variabel X (Fitur/Atribut) dan Variabel y (Kelas/Label), Memisahkan data menjadi data latih dan data uji, Inisiasi model, *Training model* dengan *.fit(),* Prediksi pada *data test*, dan Memeriksa antara hasil prediksi dan *data actual.*

```
X= data.iloc[:, :-1]
y= data.values[:, -1]

pd.DataFrame(y).head()
```

	0
0	0.0
1	0.0
2	1.0
3	0.0
4	0.0

Gambar 3. Menentukan Variabel X dan Y

Pada Gambar 3, ditampilkan kode atau sintaks untuk *data pre-processing*. Pemrosesan data dilakukan Menentukan Variabel X (Fitur/Atribut) dan Variabel y (Kelas/Label), Variabel X adalah fitur-fitur dari dataset data, menggunakan metode Pandas untuk memilih semua baris, mengambil semua kolom kecuali kolom terakhir dan kolom label. Variabel y adalah target atau label dari dataset data. menggunakan metode *.values* dari Pandas untuk mengonversi *DataFrame* menjadi *array NumPy*, kemudian memilih kolom terakhir (-1), yang merupakan kolom label.

```
[ ] # Memisahkan data menjadi data latih dan data uji
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 4. Memisahkan data menjadi data latih dan data uji

Pada Gambar 4. Ditampilkan *kode X_train, X_test, y_train, y_test:* Variabel ini akan menampung data latih dan data uji untuk fitur (X) dan label (y) secara terpisah. *train_test_split(X, y, test_size=0.2, random_state=42):* Fungsi ini membagi dataset menjadi data latih dan data uji. Argumen X adalah fitur-fitur dataset, y adalah labelnya. *test_size=0.2* menentukan bahwa 20% dari data akan digunakan sebagai data uji, sedangkan 80% sisanya akan digunakan sebagai data latih. *random_state=42* digunakan untuk membuat pembagian data yang konsisten ketika kode dieksekusi ulang.

```
# Inisiasi Model
model = BernoulliNB()
```

Gambar 5. Inisiasi Model

Pada Gambar 5. Kode di atas menginisialisasi model klasifikasi Naive Bayes dengan jenis *Bernoulli* menggunakan kelas *BernoulliNB()* dari *scikit-learn*. Ini membuat objek model yang siap untuk dilatih dengan data. Model ini cocok untuk digunakan ketika fitur-fitur input adalah variabel biner (bernilai 0 atau 1), yang sesuai dengan distribusi *Bernoulli*.

```
# Training model dengan .fit()
model.fit(X_train, y_train)
```

```
▾ BernoulliNB
BernoulliNB()
```

Gambar 6. *Training Model*

Pada Gambar 6. Kode di atas melakukan pelatihan model *machine learning* menggunakan metode *fit()* dari model yang telah diinisialisasi sebelumnya. Variabel *X_train* adalah fitur dari data latih, dan *y_train* adalah labelnya.

Dengan memanggil metode *fit()* pada model, model dilatih untuk mempelajari pola-pola yang terkandung dalam data latih, sehingga nantinya dapat digunakan untuk membuat prediksi atau klasifikasi pada data baru

```
y_pred = model.predict(X_test)
y_pred
```

```
array([1., 0., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 0.,
       0., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
       1., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0.,
       1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 1., 1., 1.,
       1., 1., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0.,
       0., 0., 1., 1., 0., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1.,
       1., 0., 0., 0., 1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 0., 0., 0.,
       1., 0., 0., 0., 0., 0., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1.,
       0., 1., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0., 1., 0.,
       0., 1., 1., 1., 0., 1., 0., 1., 0., 1., 1., 1., 0., 1., 0., 0., 1.,
       1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1.,
       1., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 1., 1., 0.,
       1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1., 1., 1., 1.,
       0., 1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 1., 0., 0., 0., 0., 0.,
       1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 0., 1., 0., 1., 0., 0., 1.,
       0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1., 1.,
       0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1.])
```

```
y_test
```

```
array([1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0.,
       0., 1., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 0., 1., 1., 1.,
       0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0.,
       0., 1., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1.,
       1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 1., 1., 1., 1., 0., 1.,
       0., 0., 1., 0., 1., 1., 0., 1., 0., 0., 0., 0., 1., 0., 1., 1., 1.,
       0., 1., 0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 0., 1., 0., 1.,
       0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1.,
       0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 1., 0., 1., 0.,
       0., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 1., 0., 1., 0., 1., 0.,
       1., 1., 1., 0., 0., 0., 1., 0., 1., 1., 0., 0., 0., 0., 0., 0., 1.,
       1., 1., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0., 0., 1., 0., 0.,
       0., 0., 0., 0., 1., 1., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 1.,
       0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 1., 0., 1.,
       0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 1.,
       1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0.,
       0., 0., 0., 0., 0., 0., 1., 0., 1., 1., 1., 0., 1.])
```

Gambar 7. Prediksi pada *data test*

Pada Gambar 7. Kode di atas menggunakan model yang telah dilatih sebelumnya untuk membuat prediksi pada data uji. Ini dilakukan dengan menggunakan metode *predict()* dari model yang telah diinisialisasi sebelumnya, dengan memberikan fitur dari data uji (*X_test*) sebagai argumennya. Hasil prediksi disimpan dalam variabel *y_pred*, yang berisi prediksi label untuk setiap sampel dalam data uji.

```
df = pd.DataFrame({'Prediksi': y_pred, 'Aktual': y_test})
df
```

	Prediksi	Aktual
0	1.0	1.0
1	0.0	0.0
2	1.0	0.0
3	0.0	0.0
4	0.0	1.0
...	...	...
280	1.0	1.0
281	1.0	1.0
282	1.0	1.0
283	0.0	0.0
284	1.0	1.0

285 rows × 2 columns

Gambar 8. Hasil Prediksi

Pada Gambar 8. Kode di atas membuat *DataFrame* baru menggunakan Pandas dengan dua kolom, yaitu 'Prediksi' yang berisi hasil prediksi dari model (*y_pred*) dan 'Aktual' yang berisi label aktual dari data uji (*y_test*). Ini berguna untuk membandingkan prediksi yang dihasilkan oleh model dengan nilai sebenarnya dalam data uji, sehingga memungkinkan untuk mengevaluasi kinerja model dengan lebih baik.

3.3 Evaluasi Model

Evaluasi model dalam penelitian ini dilakukan dengan menggunakan *confusion matrix*, yang mencakup pengukuran terhadap nilai akurasi, presisi, *recall*, dan *F1-Score.*

```
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
conf_matrix
```

```
array([[93, 64],
       [43, 85]])
```

Gambar 9. *Confusion Matrix*

Pada Gambar 9. Kode di atas menghitung matriks kebingungan (*confusion matrix*) menggunakan fungsi *confusion_matrix()* dari Scikit-Learn. Matriks kebingungan digunakan untuk mengevaluasi kinerja model klasifikasi dengan membandingkan nilai aktual (*y_test*) dengan nilai prediksi (*y_pred*). Hasilnya adalah sebuah matriks berukuran 2x2 (untuk masalah klasifikasi biner) yang berisi empat nilai: true negative (TN), false positive (FP), *false negative* (FN), dan *true positive* (TP). Matriks ini memungkinkan untuk mengevaluasi performa model dalam mengklasifikasikan sampel ke dalam kelas positif dan negatif. Dengan menggunakan data dari *confusion matrix*, performa model dapat dievaluasi melalui perhitungan akurasi, presisi, *recall,* dan *F1-score*. Berikut adalah perhitungan performa model berdasarkan tabel *classifier* pada **tabel 1.**

1. Akurasi

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN}\ x\ 100\%$$

$$\text{Akurasi} = \frac{93 + 85}{93 + 85 + 43 + 64}\ x\ 100\%$$

$$\text{Akurasi} = 0{,}62456140351\ x\ 100\% = \mathbf{62}\%$$

2. Presisi

$$\text{Presisi} = \frac{TP}{TP + FP}$$

$$\text{Presisi} = \frac{93}{93 + 43} = \mathbf{0{,}68}$$

3. *Recall*

$$Recall = \frac{TP}{TP + FN}$$

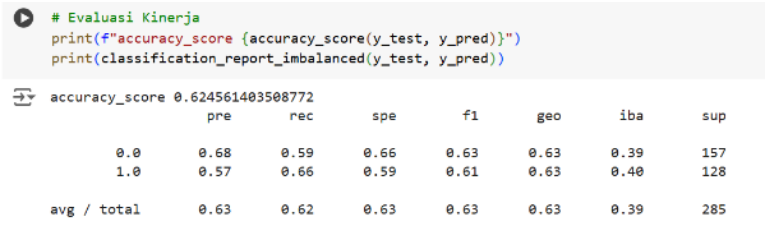
$$Recall = \frac{93}{93 + 64} = \mathbf{0{,}59}$$

4. *F1-Score*

$$F1\text{-}Score = 2\ x\ \frac{Presisi\ x\ Recall}{Presisi + Recall}$$

$$F1\text{-}Score = 2\ x\ \frac{(0{,}68\ x\ 0{,}59)}{0{,}68 + 0{,}59} = \mathbf{0{,}63}$$

Berdasarkan evaluasi hasil klasifikasi menggunakan algoritma Navie Bayes, diperoleh nilai akurasi sebesar 62%, presisi 0,68, *recall* 0,59, dan *f1-score* 0,63. Penelitian ini fokus pada penggunaan algoritma Naïve Bayes untuk klasifikasi penyakit anemia dengan data atribut terbatas. Meskipun akurasi model Naïve Bayes mencapai 62%, nilai ini masih rendah karena kesederhanaannya yang mungkin tidak cukup untuk menangani kompleksitas data yang terbatas. Untuk meningkatkan akurasi, bisa dilakukan teknik pemrosesan data lebih lanjut, memperluas dataset, atau mencoba algoritma yang lebih kompleks seperti *Decision Trees, Random Forest,* atau SVM. Penelitian ini menekankan bahwa meskipun Naïve Bayes lebih sederhana, algoritma ini tetap efektif dan efisien untuk aplikasi dunia nyata dengan data terbatas. Selain akurasi, penelitian ini juga mengevaluasi metrik lain seperti presisi dan *recall,* yang penting untuk memastikan diagnosis anemia yang akurat dan mendeteksi sebanyak mungkin kasus anemia. Presisi tinggi menunjukkan bahwa sebagian besar hasil positif yang diprediksi adalah anemia, penting untuk mencegah diagnosis salah pada pasien sehat. *Recall* yang baik berarti sebagian besar kasus anemia dapat terdeteksi, penting untuk mencegah kondisi yang tidak terdiagnosis.Rincian perhitungan tersebut dapat dilihat pada Gambar 11.



```
# Evaluasi Kinerja
print(f"accuracy_score {accuracy_score(y_test, y_pred)}")
print(classification_report_imbalanced(y_test, y_pred))
```

```
accuracy_score 0.6245614035087719
                   pre       rec       spe        f1       geo       iba       sup

        0.0       0.68      0.59      0.66      0.63      0.63      0.39       157
        1.0       0.57      0.66      0.59      0.61      0.63      0.40       128

avg / total       0.63      0.62      0.63      0.63      0.63      0.39       285
```

Gambar 11. Hasil Performa Model

4. KESIMPULAN

Setelah melakukan analisa terhadap dataset penyakit anemia menggunakan algoritma NAVIE BAYES dimana didalam dataset tersebut terdapat 1536 data pasien dengan 6 atribut, dapat disimpulkan akurasi model sebesar 0.62 atau 62.45%, menunjukkan bahwa model memiliki tingkat keberhasilan yang moderat dalam memprediksi keberadaan atau tidaknya penyakit anemia. *Precision* (pre) dan *recall* (rec) mengukur kualitas prediksi model, khususnya dalam membedakan antara kelas positif (1.0, mengindikasikan keberadaan penyakit anemia) dan kelas negatif (0.0, mengindikasikan ketiadaan penyakit anemia). *Precision* untuk kelas 0.0 adalah 0.68, sedangkan untuk kelas 1.0 adalah 0.57. Hal ini menunjukkan bahwa model cenderung lebih baik dalam memprediksi kelas 0.0 (tidak ada anemia) daripada kelas 1.0 (ada anemia). *Recall* untuk kelas 0.0 adalah 0.59, sedangkan untuk kelas 1.0 adalah 0.66. Ini menunjukkan bahwa model cenderung lebih baik dalam mendeteksi kasus anemia (kelas 1.0) daripada kasus tanpa anemia (kelas 0.0). *Support* mengindikasikan jumlah masing-masing kelas dalam data uji. Dalam konteks ini, terdapat 157 sampel tanpa anemia (kelas 0.0) dan 128 sampel dengan anemia (kelas 1.0). Secara keseluruhan, model Naive Bayes menunjukkan kinerja yang cukup baik dalam memprediksi keberadaan atau tidaknya penyakit anemia, meskipun terdapat kecenderungan untuk lebih baik dalam memprediksi kelas tanpa anemia daripada kelas dengan anemia. Evaluasi lebih lanjut dan peningkatan model mungkin diperlukan untuk meningkatkan kinerja prediksi kelas dengan anemia. Untuk penelitian masa depan, disarankan untuk mengeksplorasi penggunaan algoritma hybrid yang menggabungkan kekuatan Naïve Bayes dengan algoritma lain seperti *Decision Tree* atau SVM untuk meningkatkan akurasi tanpa mengorbankan efisiensi. Selain itu, teknik penyeimbangan data seperti SMOTE (*Synthetic Minority Over-sampling Technique*) atau penyesuaian bobot kelas dapat diterapkan untuk mengatasi masalah ketidakberimbangan kelas dalam dataset medis, yang sering kali mempengaruhi kinerja model klasifikasi.

DAFTAR PUSTAKA

[1] World Health Organization. (2020). Global Anemia Report. Geneva: WHO Press. dan Smith, J., & Zhang, L. (2021). Epidemiology of Anemia and Its Impact on Global Health. Journal of Medical Science, 25(3), 456-468.
[2] R. Susanti and L. Elfianti, "Prediksi Diagnosis Anemia Melalui Metode Klasifikasi Berbasis Machine Learning," J. Artif. Intell. Appl. (JAIA) P-ISSN xxxx-xxxx, vol. 1, no. 1, pp. 36–43, 2024. [3] F. A. Purnomo, N. M. Yoeseph, B. K. Riasti, and R. W. Anggara, "Water level detector for early warning systems using color difference measurement," in *AIP Conference Proceedings*, 2018, vol. 1977.
[3] F. K. Wardhani, N. Kamilatutsaniya, A. Alamsyah, E. Daniati, and A. Ristyawan, "Perbandingan Algoritma Naive Bayes Dan Desicion Tree Dalam Pengujian Data Anemia Menggunakan," Inotek, vol. 8, 2024, [Online]. Available: https://proceeding.unpkediri.ac.id/index.php/inotek/.
[4] Y. Maulina, A. Gunaryati, and R. T. Aldisa, "Sistem Pakar Diagnosis Awal Penyakit Anemia Menggunakan Metode Naïve Bayes dan Certainty Factor," STRING (Satuan Tulisan Ris. dan Inov. Teknol., vol. 8, no. 1, p. 110, 2023, doi: 10.30998/string.v8i1.16468.
[5] "Apa itu Algoritma Naive Bayes? Pengertian dan contoh 2024 | RevoU," Revou.co,2024. https://revou.co/kosakata/algoritma-naive-bayes#:~:text=menjadi%200%2C5.-,Manfa at%20Algoritma%20Naive%20Bayes,hidung%2C%20mulut%2C%20dan%20alis.
[6] "View of SISTEM PAKAR UNTUK DIAGNOSIS PENYAKIT ANEMIA MENGGUNAKAN TEOREMA BAYES," Unsoed.ac.id, https://jutif.if.unsoed.ac.id/index.php/jurnal/article/view/12/4.
[7] emreiekyurt, "Anemia Classification with EDA (100% Acc)," Kaggle.com, Sep.09,2022. https://www.kaggle.com/code/emreiekyurt/anemia-classification-with-eda-100-acc/inp ut
[8] P. Studi, S. Informasi, and F. I. Komputer, "SI163-MACHINE LEARNING."
[9] "View of SISTEM PAKAR UNTUK DIAGNOSIS PENYAKIT ANEMIA MENGGUNAKAN TEOREMA BAYES," Unsoed.ac.id, 2024. https://jutif.if.unsoed.ac.id/index.php/jurnal/article/view/12/4.
[10] emreiekyurt, "Anemia Classification with EDA (100% Acc)," Kaggle.com, Sep.09,2022. https://www.kaggle.com/code/emreiekyurt/anemia classification-with-eda-100-acc/inp
[11] "Apa itu Algoritma Naive Bayes? Pengertian dan contoh 2024 | RevoU," Revou.co,2024. https://revou.co/kosakata/algoritma-naive-bayes#:~:text=menjadi%200%2C5. Manfaat%20Algoritma%20Naive%20Bayes,hidung%2C%20mulut%2C%20d an%20alis.
[12] A. Sauddin and R. Ibnas, "Klasifikasi Penderita Penyakit Anemia dengan Metode NBC menggunakan R Programming," J. MSA (Mat. dan Stat. serta Apl., vol. 11, no. 2, pp. 112–123, 2024, doi: 10.24252/msa.v11i2.36470.
[13] R. Indrayani, "Analisa Perbandingan Algoritme Naive Bayes Dan Decision Tree Pada Klasifikasi Data Transfusi Darah," J. Ilm. Teknol. Infomasi Terap., vol. 5, no. 1, pp. 38–44, 2019, doi: 10.33197/jitter.vol5.iss1.2018.251.
[14] E. Maulid, "Elfina Maulid dan Diajeng Prihatin Karunia Esa Sistem Pakar Identifikasi Penyakit Anemia Menggunakan Metode Certainty Factor," vol. 2, no. 02, pp. 243–252, 2020.
[15] A. Nada, "Aplikasi Sistem Pakar Diagnosa Penyakit Anemia Menggunakan Metode Forward Chaining," JEKIN - J. Tek. Inform., vol. 1, no. 3, pp. 10–19, 2023, doi: 10.58794/jekin.v1i3.356.