

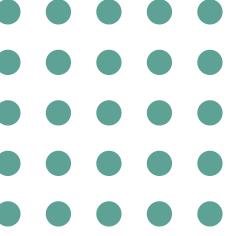
IMPLEMENTASI METODE GAUSSIAN NAÏVE BAYES UNTUK MEMBANGUN MODEL PREDIKSI SERANGAN JANTUNG

Muhammad imam ghozali

NIM : 200605110154

Latar Belakang

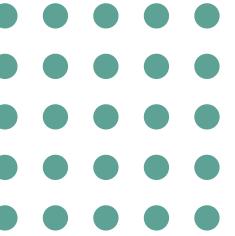
- Menurut Data WHO Penyakit jantung menyebabkan lebih dari 17 juta kematian global pada 2015 (31% dari total kematian) .
- Faktor risiko: akses layanan kesehatan, pola makan, gaya hidup, dan lingkungan.
- RISKESDAS 2018: 15 dari 1.000 orang di Indonesia menderita penyakit jantung.
- Banyak kasus ditemukan di stadium lanjut akibat kurangnya kesadaran dan pemeriksaan kesehatan rutin



Rumusan Masalah



Seberapa tinggi nilai accuracy, precision, recall, dan f1-score metode Gaussian naïve Bayes dalam memprediksi penyakit serangan jantung ?



Batasan Masalah



Data yang digunakan adalah data public yang diperoleh melalui Elsevier Mendeley Data
Repository: Heart Attack Dataset

Tujuan Penelitian

Tujuan utama dalam penelitian ini untuk mengukur tingkat akurasi penggunaan metode gaussian naive bayes dalam sistem prediksi penyakit serangan jantung berdasarkan faktor resiko.

Manfaat Penelitian

01.

Diharapkan dapat memberikan manfaat dgn adanya sistem prediksi penyakit serangan jantung dengan tingkat akurasi yang sesuai.

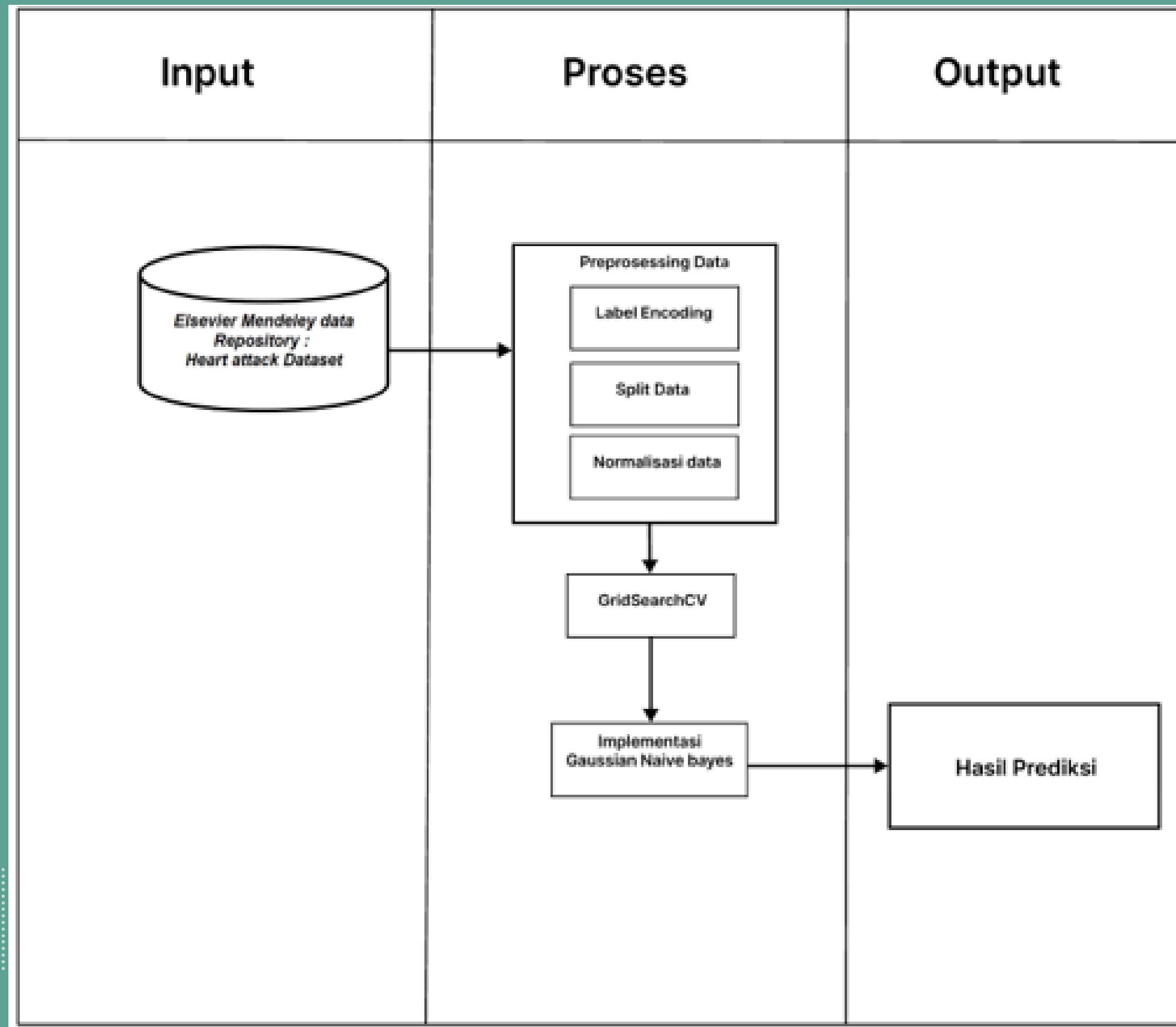
02.

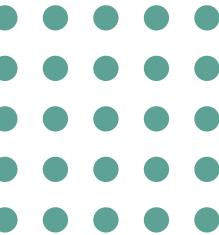
Dapat memberikan pengetahuan dalam implementasi metode Gaussian naïve bayes untuk prediksi penyakit serangan jantung.

03.

Dapat dijadikan landasan untuk penelitian selanjutnya

Desain Sistem



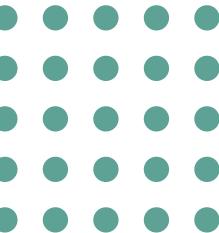


Pengumpulan Dataset



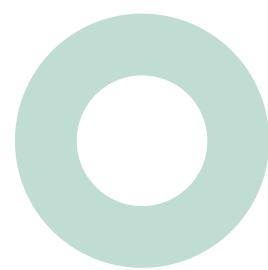
Tabel 3. 2 Contoh Dataset

Age	Gender	Heart rate	Systolic blood pressure	Diastolic blood pressure	Blood sugar	CK-MB	Troponin	Result
64	1	66	160	83	160.0	1.80	0.012	negative
21	1	94	98	46	296.0	6.75	1.060	positive
55	1	64	160	77	270.0	1.99	0.003	Negative
64	1	70	120	55	270.0	13.87	0.122	Positive
55	1	64	112	65	300.0	1.08	0.003	Negative
58	0	61	112	58	87.0	1.83	0.004	Negative



Tahap Preprocessing Data

Transformasi Data



```
#mengubah kolom kategoris menjadi kolom numerik
label_encoder = LabelEncoder()
label_encoder.fit(df["Result"])
class_mapping = dict(zip(label_encoder.classes_,
                         label_encoder.transform(label_encoder.classes_)))
print(class_mapping)
df["Result"] = label_encoder.transform(df["Result"])
```

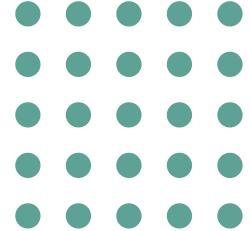
pada tahapan ini dilakukan transformasi data menggunakan Teknik LabelEncoder mengubah data kategorikal menjadi data numerik, yang bertujuan untuk membuatnya lebih mudah dipahami dan diakses oleh model Machine Learning. seperti mengubah kelas "Result" menjadi Negative adalah 0 dan positive adalah 1.

Tabel 3. 3 Contoh dataset yang belum menjalani proses transformasi data

<i>Systolic BP</i>	<i>Diastolic BP</i>	<i>Blood Sugar</i>	<i>CK-MB</i>	<i>Troponin</i>	<i>Result</i>
160	83	160.0	1.80	0.012	<i>negative</i>
214	82	87.0	300.0	2.370	<i>positive</i>

Tabel 3. 4 Contoh dataset yang telah menjalani proses transformasi data

<i>Systolic BP</i>	<i>Diastolic BP</i>	<i>Blood Sugar</i>	<i>CK-MB</i>	<i>Troponin</i>	<i>Result</i>
160	83	160.0	1.80	0.012	0
214	82	87.0	300.0	2.370	1



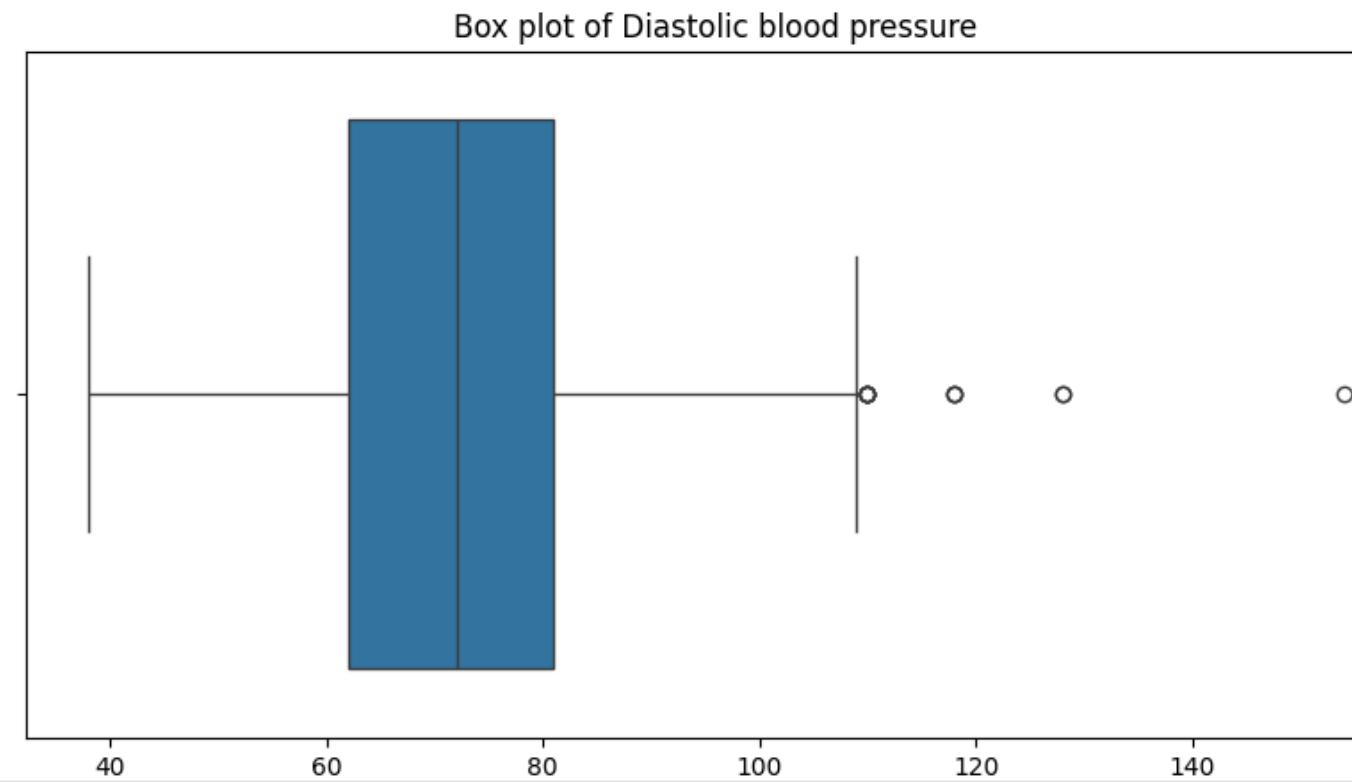
Tahap Preprocessing Data

Mendeteksi adanya data outliers

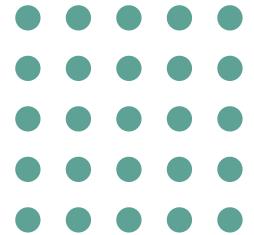
```
# Plot Boxplot for Outliers Detection
def plot_boxplot(data, column):
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=data[column])
    plt.title(f'Box plot of {column}')
    plt.show()
```

```
# Plot boxplot for each numeric feature
numeric_features = ['Age', 'Heart rate', 'Systolic blood pressure', 'Diastolic blood pressure', 'Blood sugar']
for feature in numeric_features:
    plot_boxplot(df, feature)
```

Disini saya melakukan deteksi untuk mencari nilai data outliers menggunakan plot_boxplot. Fungsi boxplot sendiri adalah untuk menggambarkan distribusi data dan mengidentifikasi outlier pada setiap fitur. Contoh hasil :



pada box plot dalam fitur Diastolic blood pressure teridentifikasi bulatan" kecil yang merupakan data outliers.



Tahap Preprocessing Data

seperti yang sudah dijelaskan sebelumnya Outliers adalah nilai yang sangat berbeda dari data lainnya dalam suatu dataset. Outliers bisa sangat tinggi atau sangat rendah dibandingkan dengan nilai-nilai rata-rata di dataset. hal ini bisa mempengaruhi kinerja model seperti :

- Akurasi tidak akurat atau overfitting (model terlalu cocok dengan data pelatihan, tetapi buruk di data baru)
- Dapat menyebabkan statistik deskriptif seperti mean (rata-rata), standar deviasi, dan varian, sehingga hasil analisis menjadi tidak representatif.
- Data outliers yang ekstrem bisa memperlambat proses pelatihan model karena algoritma mencoba untuk menyesuaikan dengan nilai-nilai tersebut.

```
# Remove Outliers using Z-score
for feature in numeric_features:
    df = df[(np.abs(zscore(df[feature])) < 3)]
```

jadi untuk menghindari resiko tersebut saya akan melakukan penghapusan outliers dengan menggunakan Z-Score agar nantinya pembagian data latih dan data uji menjadi sama baik dan stabil serta membantu mempercepat proses pelatihan tanpa mengorbankan akurasi model.

Split Data

Tahap Preprocessing Data

```
# Menentukan fitur (X) dan target (y)
x = df.drop('Result', axis=1) # Fitur
y = df['Result'] # Target

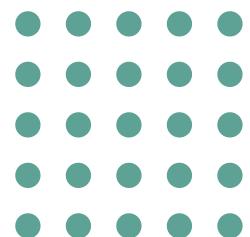
# Melakukan split data
# Model A (90:10)
x_train_A, x_test_A, y_train_A, y_test_A = train_test_split(x, y, test_size=0.1, random_state=42)
print("Model A:")
print("Jumlah data train:", len(x_train_A))
print("Jumlah data test:", len(x_test_A))

# Model B (80:20)
x_train_B, x_test_B, y_train_B, y_test_B = train_test_split(x, y, test_size=0.2, random_state=42)
print("\nModel B:")
print("Jumlah data train:", len(x_train_B))
print("Jumlah data test:", len(x_test_B))

# Model C (70:30)
x_train_C, x_test_C, y_train_C, y_test_C = train_test_split(x, y, test_size=0.3, random_state=42)
print("\nModel C:")
print("Jumlah data train:", len(x_train_C))
print("Jumlah data test:", len(x_test_C))

# Model D (65:35)
x_train_D, x_test_D, y_train_D, y_test_D = train_test_split(x, y, test_size=0.35, random_state=42)
print("\nModel D:")
print("Jumlah data train:", len(x_train_D))
print("Jumlah data test:", len(x_test_D))
```

- Split Data adalah proses membagi dataset menjadi dua subset yang terpisah, menjadi data latih (train data) dan data uji (test data). ini dilakukan dengan memisahkan data menjadi fitur (X) dan target (y). fitur (X) adalah variabel input yang digunakan untuk memprediksi nilai target. dan kolom "Result" sebagai target (y) yang ingin diprediksi.
Setelah pemisahan fitur (X) dan target (y), data dibagi menjadi dua set terpisah: data latih dan data uji menggunakan train-test split.
- Proporsi data uji dapat ditentukan dengan test_size, yang biasanya merupakan persentase dari keseluruhan dataset. Hal ini memungkinkan kita untuk mengalokasikan sebagian data untuk pengujian model, sementara sisa data digunakan untuk melatih model.



Normalisasi Data

Tahap Preprocessing Data

```
# Melakukan normalisasi data (standard scaling) terhadap fitur-fitur numerik
scaler = StandardScaler()
x_train_A_scaled = scaler.fit_transform(x_train_A)
x_test_A_scaled = scaler.transform(x_test_A)
x_train_B_scaled = scaler.fit_transform(x_train_B)
x_test_B_scaled = scaler.transform(x_test_B)
x_train_C_scaled = scaler.fit_transform(x_train_C)
x_test_C_scaled = scaler.transform(x_test_C)
x_train_D_scaled = scaler.fit_transform(x_train_D)
x_test_D_scaled = scaler.transform(x_test_D)
```

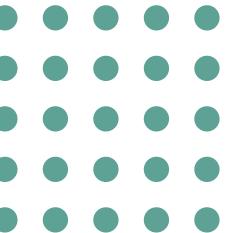
Dalam hal ini, StandardScaler adalah metode yang paling umum digunakan untuk scaling. Proses ini mengubah setiap fitur sehingga memiliki deviasi standar satu dan rata-rata nol. Dengan cara menghitung nilai rata-rata (mean) dan standard deviation (standard deviation) dari setiap fitur pada data latih. lalu nilai masing-masing fitur kemudian akan diubah dengan mengurangi nilai rata-ratanya dan membaginya dengan deviasi standar. Dengan cara ini, semua fitur akan memiliki rentang yang sebanding, yang membuat proses lebih mudah.

Tabel 3. 5 Contoh dataset yang belum menjalani normalisasi

<i>Systolic BP</i>	<i>Diastolic BP</i>	<i>Blood Sugar</i>	<i>CK-MB</i>	<i>Troponin</i>
160	83	160.0	1.80	0.012
112	58	87.0	183.0	0.004

Tabel 3. 6 Contoh dataset yang sudah menjalani proses normalisasi

<i>Systolic BP</i>	<i>Diastolic BP</i>	<i>Blood Sugar</i>	<i>CK-MB</i>	<i>Troponin</i>
1.26	0.76	0.11	-0.59	-0.40
-12.07	-18.65	-516.43	-1041.81	185.47



GridSearch CV

```
# Menggunakan GridSearchCV untuk mencari parameter terbaik untuk Model A
param_grid_A = {'var_smoothing': np.logspace(0, -9, num=100)}
nb_model_A = GaussianNB()
grid_search_A = GridSearchCV(nb_model_A, param_grid_A, cv=10)
grid_search_A.fit(x_train_A, y_train_A)

# Menampilkan parameter terbaik Model A
print("Parameter terbaik Model A:", grid_search_A.best_params_)

# Menampilkan skor validasi silang terbaik Model A
print("Skor validasi silang terbaik Model A:", grid_search_A.best_score_)

# Menampilkan model terbaik Model A
best_model_A = grid_search_A.best_estimator_
print("Model terbaik Model A:")
print(best_model_A)
```

GridSearchCV atau Grid Search Cross-Validation adalah metode yang digunakan dalam machine learning untuk mengoptimalkan hyperparameter model dengan cara yang sistematis dan efisien.

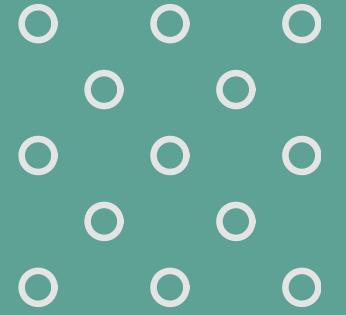
Pada kode program diatas GridSearchCV bekerja dengan terlebih dahulu mendefinisikan rentang nilai hyperparameter yang ingin diuji, seperti param_grid_A = {'var_smoothing': np.logspace(0, -9, num=100)}, yang mencakup 100 nilai dari 1 hingga 10^{-9} .

Kemudian model Gaussian Naive Bayes diinisialisasi dengan nb_model_A = GaussianNB(), kemudian GridSearchCV diatur menggunakan model tersebut dengan 10-fold cross-validation melalui grid_search_A = GridSearchCV(nb_model_A, param_grid_A, cv=10). Proses pencarian dimulai dengan membagi data latih menjadi 10 subset, selanjutnya melatih model pada 9 subset dan menguji pada subset ke-10, diulangi 10 kali untuk setiap kombinasi hyperparameter.

Kemudian rata-rata skor validasi untuk setiap kombinasi dihitung, dan kombinasi dengan skor tertinggi dipilih sebagai yang terbaik.

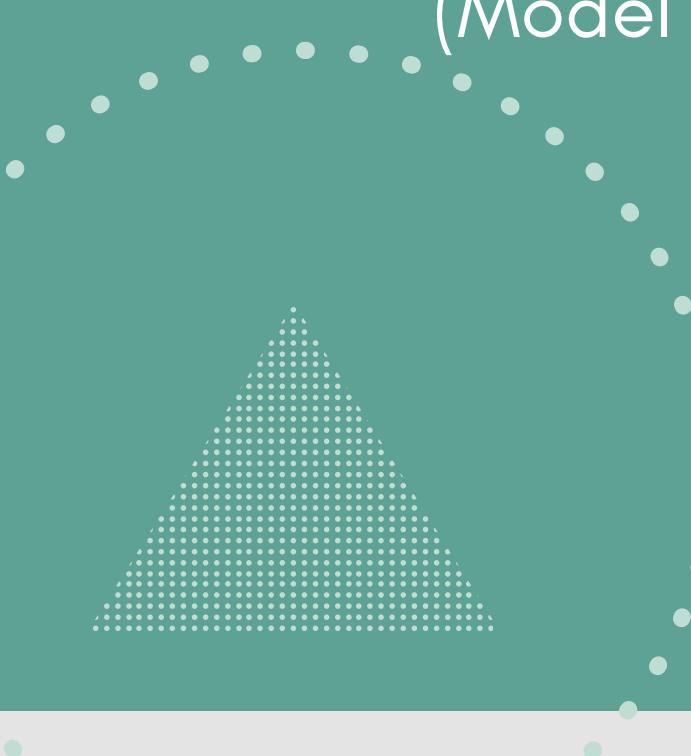
Hasilnya ditampilkan dengan menggunakan _search_A.best_params_ untuk parameter terbaik, grid_search_A.best_score_ untuk menampilkan skor validasi tertinggi, grid_search_A.best_estimator_ untuk menampilkan model terbaik.

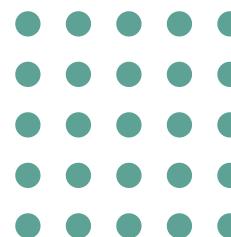
Proses ini memastikan bahwa model yang dihasilkan adalah yang paling optimal berdasarkan evaluasi menyeluruh



Hasil Uji Coba Rasio Pembagian Data

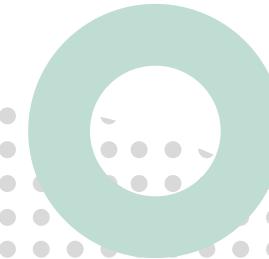
Untuk melakukan uji coba, Data dibagi dengan menggunakan train-test split dengan 4 Model. pertama (Model A) menggunakan pembagian 90% data latih dan 10% data uji, (Model B) menggunakan perbandingan 80% data latih dan 20% data uji. (Model C) membagi data dengan perbandingan 70% data latih dan 30% data uji, sedangkan (Model D) menggunakan 65% data latih dan 35% data uji.





Hasil Pengujian Model A

Rasio 90:10



```
Accuracy: 0.8677685950413223
#####
GaussianNB
#####
Number of mislabeled points out of a total 121 points : 16

precision    recall   f1-score   support
0            0.74     1.00      0.85      46
1            1.00     0.79      0.88      75

accuracy           0.87      0.87      0.87      121
macro avg       0.87     0.89      0.87      121
weighted avg    0.90     0.87      0.87      121
```

Hasil Nilai Aktual dan prediksi Model A

ID	Actual	Predicted
115	1	1
819	1	0
335	0	0
123	1	1
703	0	0
.....
1240	1	1
846	0	0
500	1	1
751	1	1
589	1	1

Class	Actual	Predicted
1	75	59
0	46	62
Jumlah nilai yang salah prediksi ada 16		

Kelas 0 (Negative) :

Precision: 0.74

Recall: 1.00

F1 Score: 0.85

Total data Pengujian: 46

Total Akurasi :

Accuracy: 0.87 atau 86.78%

Total Data Pengujian: 121

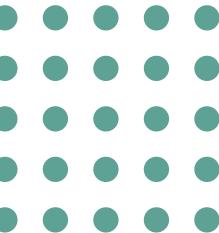
Kelas 1 (Positive) :

Precision: 1.00

Recall: 0.79

F1 Score: 0.88

Total Data Pengujian: 75



Hasil Pengujian Model B

Rasio 80:20



```
Accuracy: 0.9049586776859504
#####
GaussianNB
#####
Number of mislabeled points out of a total 242 points : 23

precision    recall   f1-score   support
0            0.81      1.00      0.89       95
1            1.00      0.84      0.92      147

accuracy          0.90      0.90      0.90      242
macro avg       0.90      0.92      0.90      242
weighted avg    0.92      0.90      0.91      242
```

Hasil Nilai Aktual dan prediksi Model B

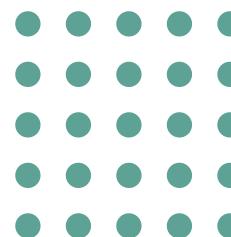
ID	Actual	Predicted
115	1	1
819	1	0
335	0	0
123	1	1
703	0	0
.....
453	0	0
479	1	0
985	1	1
1100	0	0
498	1	1

Class	Actual	Predicted
1	147	124
0	95	118
Jumlah nilai yang salah prediksi ada 23		

Kelas 0 (Negative) :
Precision: 0.81
Recall: 1.00
F1 Score: 0.89
Data Pengujian: 95

Kelas 1 (Positive) :
Precision: 1.00
Recall: 0.84
F1 Score: 0.92
Data Pengujian: 147

Total Akurasi :
Accuracy: 0.90 atau 90.50%
Total Data Pengujian: 242



Hasil Pengujian Model C

Rasio 70:30



```
Accuracy: 0.9118457300275482
#####
GaussianNB
#####
Number of mislabeled points out of a total 363 points : 32
```

	precision	recall	f1-score	support
0	0.82	1.00	0.90	141
1	1.00	0.86	0.92	222
accuracy			0.91	363
macro avg	0.91	0.93	0.91	363
weighted avg	0.93	0.91	0.91	363

```
#####
```

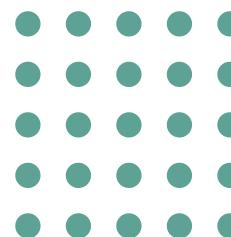
Hasil Nilai Aktual dan prediksi Model C

ID	Actual	Predicted
115	1	1
819	1	0
335	0	0
123	1	1
703	0	0
.....
410	1	1
947	1	1
792	0	0
1056	1	1
807	1	1

Class	Actual	Predicted
1	222	190
0	141	173
Jumlah nilai yang salah prediksi ada 32		

Kelas 0 (Negative) : Kelas 1 (Positive) :
Precision: 0.82 Precision: 1.00
Recall: 1.00 Recall: 0.86
F1 Score: 0.90 F1 Score: 0.92
Data Pengujian: 141 Data Pengujian: 222

Total Akurasi :
Accuracy: 0.91 atau 91.18%
Total Data Pengujian: 363



Hasil Pengujian Model D

Rasio 65:35



```
Accuracy: 0.9054373522458629
#####
GaussianNB
#####
Number of mislabeled points out of a total 423 points : 40
precision    recall   f1-score   support
0            0.80      1.00      0.89      160
1            1.00      0.85      0.92      263
accuracy          0.90      0.92      0.90      423
macro avg       0.90      0.92      0.90      423
weighted avg    0.92      0.91      0.91      423
```

Hasil Nilai Aktual dan prediksi Model D

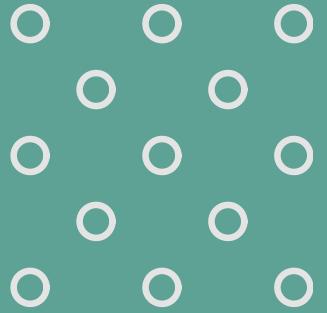
ID	Actual	Predicted
115	1	1
819	1	0
335	0	0
123	1	1
703	0	0
.....
544	0	0
798	1	0
199	1	0
1086	0	0
765	1	1

Class	Actual	Predicted
1	263	223
0	160	200
Jumlah nilai yang salah prediksi ada 40		

Kelas 0 (Negative) : Precision: 0.80
Recall: 1.00
F1 Score: 0.89
Data Pengujian: 160

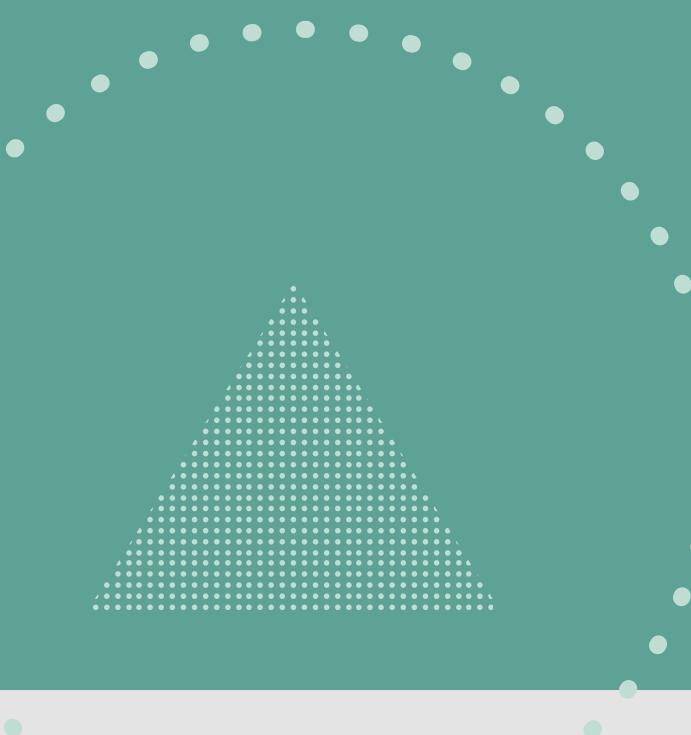
Kelas 1 (Positive) : Precision: 1.00
Recall: 0.85
F1 Score: 0.92
Data Pengujian: 263

Total Akurasi :
Accuracy: 0.91 atau 90.54%
Total Data Pengujian: 423



Hasil Uji Coba K-Fold Cross Validation

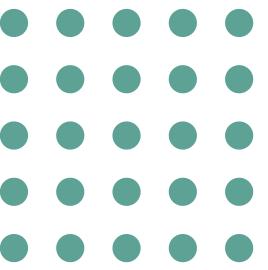
Kasus uji yang akan digunakan pada penelitian ini melibatkan metode Cross-Validation atau validasi silang dan nilai K yang digunakan adalah 5, 10, 15, dan 20. yang bertujuan untuk mengukur Kinerja model, meningkatkan akurasi, serta menghasilkan evaluasi kinerja yang konsisten.



Hasil Cross Validation K-5

Fold	Hasil Accuracy 5-fold Cross Validation
1	90.08%
2	90.91%
3	93.80%
4	90.04%
5	92.12%
Rata-rata	91.39%

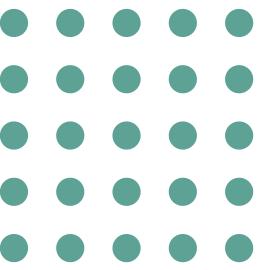
- Nilai Teritinggi didapat pada fold ketiga 93.80%.
- Nilai Terendah didapat pada fold empat 90.04%
- Perbedaan akurasi di setiap fold menunjukkan variasi performa model.
- Penurunan akurasi dapat disebabkan oleh perbedaan distribusi atau pola data di setiap lipatan.
- Model Gaussian Naive Bayes menunjukkan performa yang bervariasi di setiap fold, namun secara keseluruhan, model ini mampu memprediksi dengan sangat baik, dengan rata-rata akurasi yang cukup stabil 91.39%



Hasil Cross Validation K-10

Fold	Hasil Accuracy 10-fold Cross Validation
1	92.56%
2	88.43%
3	90.91%
4	90.08%
5	90.91%
6	96.69%
7	90.08%
8	90.08%
9	92.50%
10	91.67%
Rata-rata	91.39%

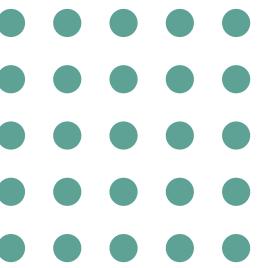
- Akurasi tertinggi sebesar 96.69% pada fold ke-6 dan terendah 88.43% pada fold ke-2.
- Model Gaussian Naive Bayes menunjukkan performa yang bervariasi di setiap fold, namun secara keseluruhan, model ini mampu memprediksi dengan sangat baik, dengan rata-rata akurasi yang cukup stabil 91.39%.



Hasil Cross Validation K-15

Fold	Hasil Accuracy 15-fold Cross Validation
1	90.12%
2	95.06%
3	86.42%
4	88.89%
5	92.59%
6	90.12%
7	90.12%
8	95.06%
9	96.25%
10	88.75%
11	88.75%
12	92.50%
13	93.75%
14	92.50%
15	90.00%
Rata-rata	91.39%

- pada fold ke-9 Akurasi tertinggi sebesar 96.25%, Akurasi terendah sebesar 88.75% pada fold ke-10 & fold ke-11.
- Rata-rata akurasi sebesar 91.39%.

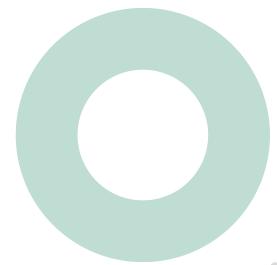


Hasil Cross Validation K-20

Fold	Hasil Accuracy 20-fold Cross Validation
1	88.52%
2	96.72%
3	88.52%
4	86.89%
5	88.52%
6	93.44%
7	91.80%
8	90.16%
9	86.67%
10	95.00%
11	96.67%
12	96.67%
13	85.00%
14	95.00%
15	85.00%
16	95.00%
17	95.00%
18	90.00%
19	95.00%
20	88.33%
Rata-rata	91.40%

- Akurasi tertinggi sebesar 96.72% pada fold ke-2, dan Akurasi terendah didapat pada fold ke-13 dan fold ke-15: sebesar 85.00%.
- Rata-rata akurasi keseluruhan: 91.40%.

Kesimpulan



Kesimpulan Hasil Pengujian Model Gaussian Naive Bayes

Pembagian Data Latih dan Uji

- Model Gaussian Naive Bayes mencapai akurasi terbaik sebesar 91% dengan rasio pembagian data 70:30.
- Evaluasi menggunakan metrik accuracy, precision, recall, dan f1-score menunjukkan hasil yang cukup baik.

K-Fold Cross Validation

- Hasil terbaik dicapai pada K=20 dengan akurasi tertinggi sebesar 96.72%.
- Rata-rata akurasi tertinggi dari semua percobaan K-fold cross validation adalah 91.40%, hal ini dikarenakan data yang dilatih dalam setiap lipatan hingga 20 terbukti menjadi faktor yang membuat

Analisis Variasi Hasil

- Variasi dalam akurasi antar fold dipengaruhi oleh distribusi data atau pola data di setiap lipatan, dan ukuran sampel.
- Meskipun ada sedikit perbedaan dalam akurasi disetiap fold, rata-rata akurasi menunjukkan bahwa model Gaussian Naive Bayes secara keseluruhan dapat memprediksi data dengan baik dan stabil yaitu disekitar 91.39% hingga 91.40% hasil yang stabil ini dipengaruhi oleh proses pengambilan sampel data yang bagus dan merata, dikarenakan sebelumnya dalam preprocessing data saya telah menghilangkan data outliers yg menyebabkan ketidakstabilan saat mengambil sampel data yang akan digunakan untuk training.



Terima Kasih

