

## **CLASSIFICATION**

disusun untuk memenuhi tugas  
mata kuliah Pembelajaran Mesin

Oleh:

**FARAH NASYWA (2208107010051)**  
**IWANI KHAIRINA (2208107010051)**  
**DINDA MAHARANI (2208107010081)**



**JURUSAN INFORMATIKA**  
**FAKULTAS MATEMATIKA DAN ILMU**  
**PENGETAHUAN ALAM**  
**UNIVERSITAS SYIAH KUALA**  
**DARUSSALAM, BANDA ACEH 2025**

1. Dataset Heart Failure Prediction ini berisi rekam medis dari 5000 pasien yang mengalami gagal jantung, dikumpulkan selama periode tindak lanjut. Tujuan dataset ini adalah untuk memprediksi apakah pasien meninggal selama masa tindak lanjut. Dataset ini berisi 13 fitur klinis yang terkait dengan kondisi kesehatan pasien.

Penjelasan tentang kolom sebagai berikut:

- 1) Age (usia): Usia pasien dalam tahun.
- 2) Anaemia (anemia): Menunjukkan apakah pasien mengalami penurunan sel darah merah atau hemoglobin, yang mempengaruhi pengangkutan oksigen.
- 3) Creatinine Phosphokinase (Kreatin kinase): Level enzim CPK dalam darah (mcg/L), yang menunjukkan kerusakan otot, khususnya otot jantung.
- 4) Diabetes: Menunjukkan apakah pasien menderita diabetes.
- 5) Ejection Fraction (Fraksi ejeksi): Persentase darah yang keluar dari jantung setiap kontraksi, dan penting untuk menilai fungsi jantung (dalam persen).
- 6) High Blood Pressure (Hipertensi): Menunjukkan apakah pasien memiliki tekanan darah tinggi.
- 7) Platelets (Trombosit): Jumlah trombosit dalam darah (kilotrombosit/mL), yang menunjukkan kemampuan pembekuan darah.
- 8) Sex (Jenis kelamin): Menunjukkan apakah pasien pria atau wanita.
- 9) Serum Creatinine (Kreatinin serum): Level kreatinin dalam darah (mg/dL), digunakan untuk menilai fungsi ginjal.
- 10) Sodium Serum (Serum Sodium): Level sodium dalam darah (mEq/L), yang penting untuk menjaga keseimbangan cairan.
- 11) Smoking (Merokok): Menunjukkan apakah pasien merokok atau tidak.
- 12) Time (Waktu): Periode tindak lanjut dalam hari.
- 13) DEATH\_EVENT (Kematian): Menunjukkan apakah pasien meninggal selama masa tindak lanjut (1 untuk meninggal, 0 untuk masih hidup).

## K-NEAREST NEIGHBOR (KNN)

### a. Import Library

```
[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

**Gambar 1.** Syntax import library

Melakukan import library pandas dan numpy untuk mengelola dan memanipulasi data juga untuk melakukan operasi pada array dan matriks multidimensi, sedangkan mengimpor library matplotlib dan seaborn untuk membuat berbagai jenis plot untuk visualisasi data. Kemudian mengimpor beberapa library dari scikit-learn yang menyediakan berbagai alat dan fungsi penting untuk membangun model machine learning, di antara nya ada LabelEncoder, StandardScaler, train\_test\_split, KNeighborsClassifier, classification\_report, confusion\_matrix, ConfusionMatrixDisplay, precision\_score dan recall\_score.

### b. Pemanggilan data

```
[9] # Pemanggilan data
datafarah= pd.read_csv('/content/heart_failure_clinical_records.csv')
datafarah
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	55.0	0	748	0	45	0	263358.03	1.3	137	1	1	88	0
1	65.0	0	56	0	25	0	305000.00	5.0	130	1	0	207	0
2	45.0	0	582	1	38	0	319000.00	0.9	140	0	0	244	0
3	60.0	1	754	1	40	1	328000.00	1.2	126	1	0	90	0
4	95.0	1	582	0	30	0	461000.00	2.0	132	1	0	50	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	45.0	0	582	1	55	0	543000.00	1.0	132	0	0	250	0
4996	60.0	1	582	0	30	1	127000.00	0.9	145	0	0	95	0
4997	95.0	1	112	0	40	1	196000.00	1.0	138	0	0	24	1
4998	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1
4999	40.0	0	244	0	45	1	275000.00	0.9	140	0	0	174	0

5000 rows x 13 columns

**Gambar 2.** Syntax dan output memanggil data

Memanggil data dengan format csv dengan membuat nama data adalah datafarah dan menghasilkan output data sebanyak 5000 baris dan 13 kolom.

c. Imputasi data

```
datafarah1 = datafarah.drop(['anaemia', 'diabetes', 'high_blood_pressure', 'sex', 'smoking'], axis=1)
```

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium	time	DEATH_EVENT
0	55.0	748	45	263358.03	1.3	137	88	0
1	65.0	56	25	305000.00	5.0	130	207	0
2	45.0	582	38	319000.00	0.9	140	244	0
3	60.0	754	40	328000.00	1.2	126	90	0
4	95.0	582	30	461000.00	2.0	132	50	1
...	...	...	...	...	...	...	...	...
4995	45.0	582	55	543000.00	1.0	132	250	0
4996	60.0	582	30	127000.00	0.9	145	95	0
4997	95.0	112	40	196000.00	1.0	138	24	1
4998	65.0	160	20	327000.00	2.7	116	8	1
4999	40.0	244	45	275000.00	0.9	140	174	0

5000 rows x 9 columns

**Gambar 3.** Syntax dan output imputasi data

Menggunakan fungsi drop dan axis=1 untuk menghapus variabel data berdasarkan kolom. Dapat dilihat ada beberapa variabel yang dihapus yaitu 'aneamia', 'diabetes', 'high\_blood\_pressure', 'sex', dan 'smoking'.

d. Data kosong

```
datafarah1.isnull().sum()
```

	0
age	0
creatinine_phosphokinase	0
ejection_fraction	0
platelets	0
serum_creatinine	0
serum_sodium	0
time	0
DEATH_EVENT	0

dtype: int64

**Gambar 4.** Syntax dan output melihat data kosong

Berdasarkan output di atas dapat diketahui bahwa tidak ada variabel yang memiliki data kosong.

e. Duplikasi data

```
[14] datafarah1.duplicated().sum()
```

3739

```
[ ] datafarah1.drop_duplicates(inplace=True)
```

**Gambar 5.** Syntax dan output melihat dan menghapus duplikasi data

Melihat duplikasi data menggunakan fungsi duplicated dan menghasilkan output duplikasi data sebanyak 3739 data, kemudian dengan fungsi drop\_duplicates untuk menghapus data yang duplikat.

f. Informasi data

```
datafarah1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   age                    5000 non-null   float64
1   creatinine_phosphokinase 5000 non-null   int64  
2   ejection_fraction      5000 non-null   int64  
3   platelets              5000 non-null   float64
4   serum_creatinine        5000 non-null   float64
5   serum_sodium            5000 non-null   int64  
6   time                   5000 non-null   int64  
7   DEATH_EVENT             5000 non-null   int64  
dtypes: float64(3), int64(5)
memory usage: 312.6 KB
```

**Gambar 6.** Syntax dan output melihat informasi data

Berdasarkan output di atas dapat diketahui bahwa setelah data duplikasi di hapus, kini data memiliki 7 kolom variabel dan sebanyak 5000 baris dengan tipe data nya ada float dan integer.

g. Statistika deskriptif data

```
datafarah1.describe()
```

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium	time	DEATH_EVENT
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	60.288736	586.760600	37.734600	265075.404370	1.369106	136.808200	130.678800	0.313600
std	11.697243	976.733979	11.514855	97999.758622	1.009750	4.464236	77.325828	0.464002
min	40.000000	23.000000	14.000000	25100.000000	0.500000	113.000000	4.000000	0.000000
25%	50.000000	121.000000	30.000000	215000.000000	0.900000	134.000000	74.000000	0.000000
50%	60.000000	248.000000	38.000000	263358.030000	1.100000	137.000000	113.000000	0.000000
75%	68.000000	582.000000	45.000000	310000.000000	1.400000	140.000000	201.000000	1.000000
max	95.000000	7861.000000	80.000000	850000.000000	9.400000	148.000000	285.000000	1.000000

**Gambar 7.** Syntax dan output statistika deskriptif data

Melihat statistika deskriptif dari data menggunakan fungsi describe dan menghasilkan output count, mean, standar deviasi, nilai minimum dan maksimum, kuartil atas, median, dan kuartil bawah untuk setiap variabel yang ada pada data.

h. Mengelompokkan data

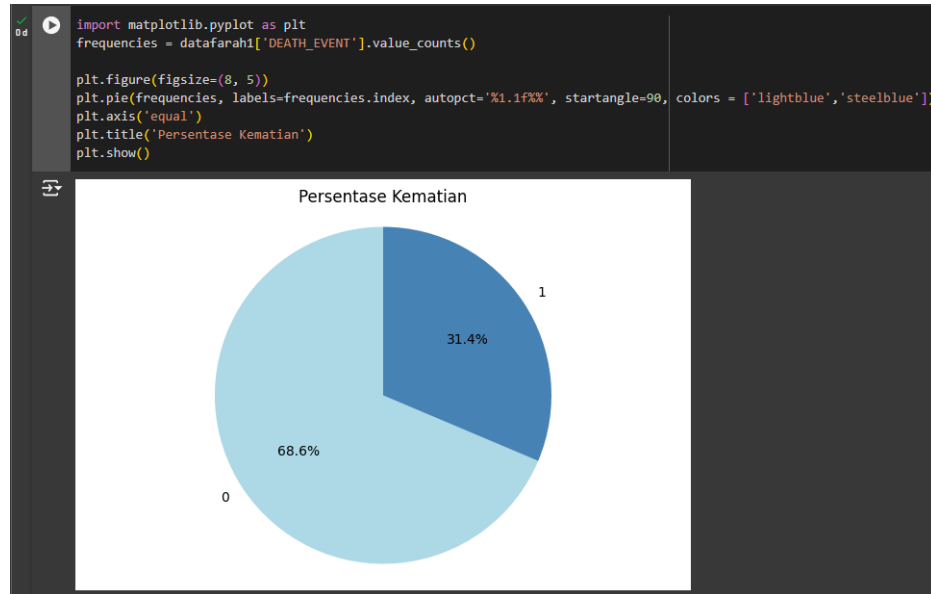
```
count_data_dependent = datafarah1.groupby("DEATH_EVENT").count()
count_data_dependent
```

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium	time
DEATH_EVENT							
0	3432		3432	3432	3432	3432	3432
1	1568		1568	1568	1568	1568	1568

**Gambar 8.** Syntax dan output mengelompokkan data

Berdasarkan output di atas, menggunakan fungsi groupby().count() berdasarkan kolom variabel DEATH\_EVENT, kita dapat mengetahui jumlah frekuensi data nya yaitu menunjukkan seberapa banyak pasien yang tidak mengalami kematian (0) dan yang mengalami kematian (1). Sehingga ada 3432 pasien yang selamat dan 1568 pasien yang meninggal selama masa tindak lanjut.

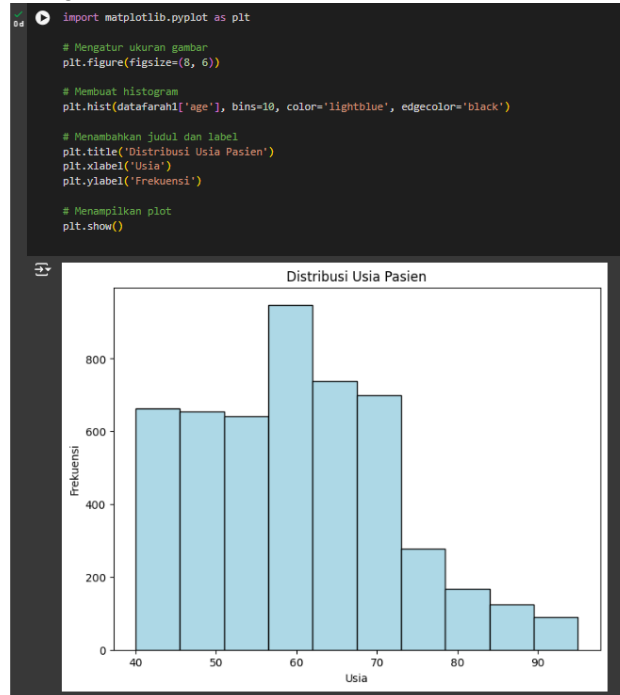
i. Visualisasi data



**Gambar 9.** Syntax dan output Pie Chart dari Death Event

Berdasarkan output pie chart dari variabel DEATH\_EVENT di atas, dapat diketahui bahwa kedua kelompok memiliki perbedaan proporsi yang cukup signifikan, di mana kejadian yang tidak mengalami kematian (68.6%) lebih besar dibandingkan dengan yang mengalami kematian (31.4%).

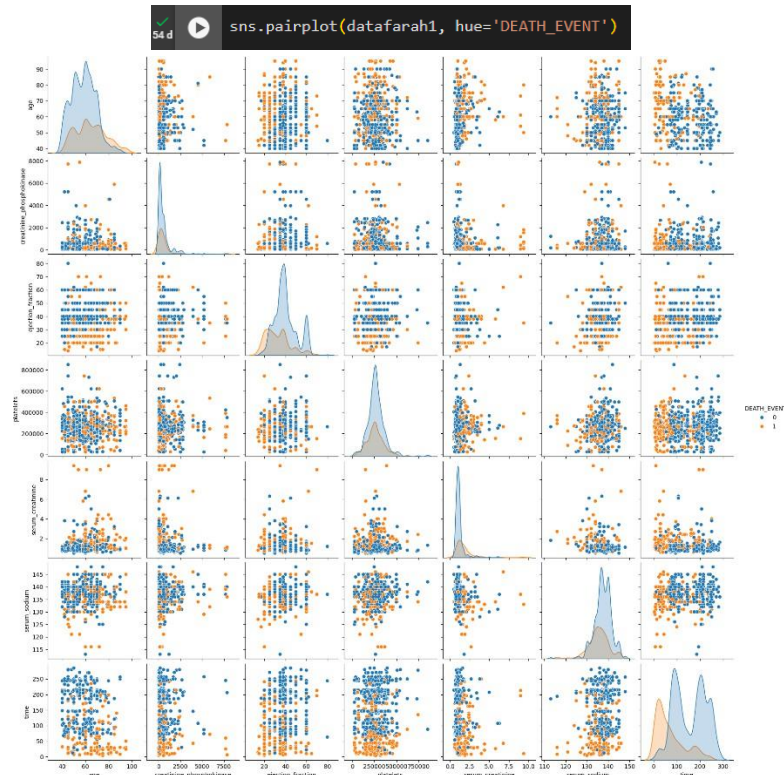
- Histogram dari Age



**Gambar 10.** Syntax dan Output Histogram dari Age

Berdasarkan output histogram dari variabel Age di atas, dapat diketahui bahwa data cenderung menjulur ke kanan (skewness to right) dengan sebaran frekuensi data terbanyak ada pada rentang usia 60-70.

- Pair Plot dari data



**Gambar 11.** Syntax dan output Pair Chart dari data

Berdasarkan output di atas dapat diketahui, dari scatter plot yang melibatkan variabel **Age**, terlihat bahwa usia memiliki hubungan yang cukup jelas dengan beberapa variabel seperti *ejection\_fraction* dan *serum\_creatinine*, di mana pasien yang lebih tua cenderung memiliki nilai yang lebih rendah pada *ejection\_fraction* dan lebih tinggi pada *serum\_creatinine*, yang berkorelasi dengan risiko kematian yang lebih besar. Namun, variabel seperti *creatinine\_phosphokinase*, *platelets*, *serum\_sodium*, dan *time* tidak menunjukkan korelasi yang signifikan dengan usia. Secara keseluruhan, pasien yang lebih tua cenderung memiliki risiko kematian yang lebih tinggi, terutama ketika dikombinasikan dengan faktor-faktor lain seperti penurunan fungsi jantung dan ginjal.

Berdasarkan scatter plot yang melibatkan **Creatinine\_Phosphokinase**, tidak ada hubungan yang jelas antara variabel ini dengan variabel lainnya seperti *age*, *ejection\_fraction*, *platelets*, *serum\_creatinine*, *serum\_sodium*, atau *time*. *Creatinine\_Phosphokinase* tampaknya tersebar secara acak di semua variabel lain, dan kematian terjadi di berbagai tingkat tanpa pola yang dapat diidentifikasi.

Dari scatter plot yang melibatkan **Ejection\_Fraction**, terlihat bahwa variabel ini memiliki hubungan yang cukup signifikan dengan beberapa variabel lain yang terkait



dengan risiko kematian. *Ejection\_fraction* yang rendah (di bawah 40%) sering dikaitkan dengan usia yang lebih tua dan kadar *serum\_creatinine* yang lebih tinggi (indikasi gangguan fungsi ginjal), yang keduanya merupakan faktor risiko penting untuk kematian. *Ejection\_fraction* rendah juga tampaknya meningkatkan kemungkinan kematian pada periode waktu yang lebih pendek. Namun, tidak terlihat hubungan yang kuat antara *ejection\_fraction* dan variabel seperti *creatinine\_phosphokinase*, *platelets*, dan *serum\_sodium*, yang berarti bahwa fungsi jantung mungkin tidak berkorelasi erat dengan enzim, jumlah trombosit, atau kadar sodium dalam memprediksi risiko kematian. Secara keseluruhan, *ejection\_fraction* merupakan variabel yang penting dalam memprediksi mortalitas, terutama jika dikombinasikan dengan usia yang lebih tua dan masalah fungsi ginjal.

Berdasarkan scatter plot yang melibatkan variabel **Platelets**, terlihat bahwa tidak ada hubungan yang signifikan antara jumlah trombosit dengan variabel lainnya seperti *age*, *creatinine\_phosphokinase*, *ejection\_fraction*, *serum\_creatinine*, *serum\_sodium*, maupun *time*. *Platelets* tampaknya tersebar secara acak di seluruh dataset dan tidak menunjukkan pola yang dapat digunakan untuk memprediksi kematian.

Dari scatter plot yang melibatkan variabel **Serum\_Creatinine**, terlihat bahwa kadar *serum\_creatinine* yang tinggi (indikasi gangguan fungsi ginjal) memiliki hubungan yang signifikan dengan beberapa variabel lain.

*Serum\_creatinine* yang tinggi lebih sering terlihat pada pasien usia lanjut dan pasien dengan *ejection\_fraction* yang rendah (penurunan fungsi jantung), dan kedua kondisi ini meningkatkan risiko kematian. *Serum\_creatinine* yang tinggi tampaknya menjadi faktor risiko penting dalam konteks mortalitas, terutama jika dikaitkan dengan penurunan fungsi jantung dan usia lanjut. Namun, tidak ada hubungan yang signifikan antara *serum\_creatinine* dengan variabel seperti *creatinine\_phosphokinase*, *platelets*, dan *serum\_sodium*, yang menunjukkan bahwa gangguan fungsi ginjal tidak selalu terkait dengan faktor-faktor ini.

Dari scatter plot yang melibatkan **Serum\_Sodium**, terlihat bahwa kadar serum sodium yang rendah berpotensi terkait dengan peningkatan risiko kematian, terutama bila dikombinasikan dengan variabel seperti *ejection\_fraction* rendah dan *serum\_creatinine* tinggi. Ini menunjukkan bahwa rendahnya kadar natrium dalam darah dapat menjadi indikasi adanya masalah jantung (penurunan fungsi jantung) dan masalah ginjal (serum

creatinine tinggi), yang bersama-sama meningkatkan risiko kematian. Namun, *serum\_sodium* tidak menunjukkan hubungan yang signifikan dengan variabel lain seperti *age*, *creatinine\_phosphokinase*, *platelets*, dan *time*, sehingga faktor-faktor tersebut tampaknya tidak mempengaruhi kadar natrium dalam darah secara signifikan dalam konteks risiko kematian.

Berdasarkan scatter plot yang melibatkan variabel **Time**, dapat disimpulkan bahwa *time* (durasi waktu sejak diagnosis atau perawatan) tidak menunjukkan hubungan yang kuat dengan variabel lain seperti *age*, *creatinine\_phosphokinase*, *ejection\_fraction*, *platelets*, *serum\_creatinine*, dan *serum\_sodium*. Kematian terjadi secara acak pada berbagai durasi waktu, dan lamanya waktu pengamatan tidak tampak mempengaruhi variabel-variabel ini secara signifikan.

Maka dari pair plot ini, dapat disimpulkan bahwa pada variabel **Age**, **Ejection Fraction**, **Serum\_Creatinine**, dan **Serum\_Sodium** memiliki perbedaan distribusi yang jelas antara pasien yang meninggal dan yang tidak. Sehingga variabel-variabel tersebut mungkin berperan lebih besar dalam memprediksi kematian dibandingkan dengan variabel lainnya.

j. Membagi dan mengubah data

[illegible]

**Gambar 12.** Syntax dan output membagi dan mengubah data

Syntax di atas digunakan untuk mempersiapkan data sebelum digunakan dalam model machine learning. Pertama, variabel  $x$  diambil dengan menghapus kolom `DEATH_EVENT`, sedangkan variabel  $y$  berisi kolom `DEATH_EVENT` yang ingin diprediksi. Data kemudian dibagi menjadi data pelatihan (80%) dan pengujian (20%) menggunakan fungsi `train_test_split`, dengan pengacakan dan stratifikasi agar distribusi kelas tetap seimbang. Selanjutnya, fungsi `LabelEncoder` digunakan untuk mengonversi variabel target biner menjadi bentuk numerik yang sesuai, meskipun dalam kasus ini sudah dalam format biner (0 dan 1). Akhirnya, data  $y_{train}$  dan  $y_{test}$  diubah dengan `transform()` untuk memastikan

keteragaman format, sehingga siap digunakan untuk melatih dan menguji model machine learning.

k. Menentukan nilai K

```
[28] scaler = StandardScaler()
      scaler.fit(X_train)

      X_train = scaler.transform(X_train)
      X_test = scaler.transform(X_test)

[29] k_list = list(range(3, 20, 2)) #k yang ganjil saja
      k_list

[3, 5, 7, 9, 11, 13, 15, 17, 19]
```

**Gambar 13.** Syntax dan output untuk menentukan nilai K

Menggunakan fungsi StandardScaler untuk menstandarisasi variabel data pelatihan dan pengujian dengan menghitung rata-rata dan standar deviasi dari X\_train, lalu menerapkan transformasi ini ke X\_train dan X\_test untuk memastikan semua fitur memiliki skala yang sama. Kemudian membuat daftar nilai k yang ganjil antara 3 hingga 19, yang akan digunakan untuk mencari nilai k terbaik dalam algoritma KNN. Pemilihan k ganjil berguna untuk menghindari kemungkinan hasil seri dalam proses klasifikasi.

l. Melakukan iterasi untuk setiap nilai K

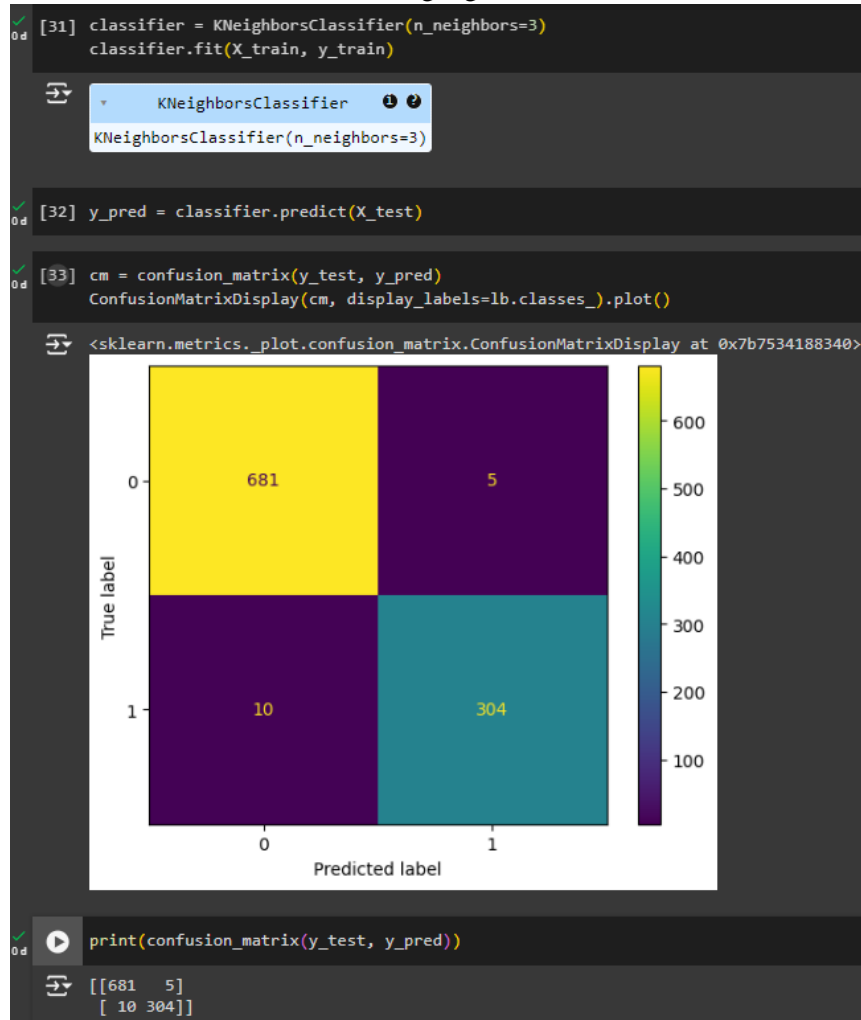
```
#lakukan iterasi untuk setiap nilai k
from sklearn.metrics import accuracy_score
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k).fit(X_train, y_train)
    y_pred = knn.predict(X_train)
    accuracy = accuracy_score(y_train, y_pred)
    print(f"K = {k}, Akurasi = {accuracy:.4f}")

K = 3, Akurasi = 0.9852
K = 5, Akurasi = 0.9798
K = 7, Akurasi = 0.9768
K = 9, Akurasi = 0.9760
K = 11, Akurasi = 0.9742
K = 13, Akurasi = 0.9715
K = 15, Akurasi = 0.9615
K = 17, Akurasi = 0.9553
K = 19, Akurasi = 0.9530
```

**Gambar 14.** Syntax dan output untuk iterasi nilai k

Menggunakan fungsi evaluate\_k\_values ini untuk menerima data pelatihan (X\_train, y\_train) dan daftar nilai k (k\_list), kemudian melakukan iterasi untuk setiap nilai k. Setelah itu model tersebut membuat prediksi untuk data pelatihan menggunakan predict(). Akurasi prediksi diukur menggunakan accuracy\_score, yang membandingkan prediksi dengan data asli di y\_train. Hasil akurasi untuk setiap nilai k kemudian ditampilkan, sehingga dapat dievaluasi mana nilai k yang memberikan hasil terbaik. Pada output di atas nilai k yang memberikan akurasi tertinggi ada pada akurasi dengan nilai k = 3.

m. Membuat model KNN dan matriks kebingungan



**Gambar 15.** Syntax dan output membuat model KNN dan matriks kebingungan

Menggunakan akurasi nilai  $k = 3$  dengan besar akurasi 0.9852 dengan model yang telah dibuat menghasilkan matriks kebingungan (confusion matrix) pada gambar di atas.

Berdasarkan output matriks kebingungan dapat diketahui pada prediksi 0 (true negative) sebanyak 681 kali model memprediksi dengan benar bahwa pasien tidak meninggal dan prediksi 0 (false negative) sebanyak 10 kali model salah memprediksi bahwa pasien tidak meninggal, padahal sebenarnya meninggal. Kemudian pada prediksi 1 (true positive) sebanyak 304 kali model memprediksi dengan benar bahwa pasien meninggal dan prediksi 1 (false negative) sebanyak 5 kali model salah memprediksi bahwa pasien meninggal, padahal sebenarnya tidak.

n. Menghitung nilai presisi, recall, dan akurasi

```
[35] # Menghitung nilai presisi, recall, dan akurasi
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

# Membuat DataFrame
metrics = {'Metric': ['Precision', 'Recall', 'Accuracy'],
          'Value': [precision, recall, accuracy]}

metrics_df = pd.DataFrame(metrics)
metrics_df
```

	Metric	Value
0	Precision	0.983819
1	Recall	0.968153
2	Accuracy	0.985000

Langkah berikutnya: [Buat kode dengan metrics\\_df](#) [Lihat plot](#)

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	686
1	0.98	0.97	0.98	314
accuracy			0.98	1000
macro avg	0.98	0.98	0.98	1000
weighted avg	0.98	0.98	0.98	1000

**Gambar 16.** Syntax dan output menghitung nilai presisi, recall, dan akurasi

Presisi digunakan untuk mengukur seberapa banyak dari prediksi positif yang benar-benar positif. Recall digunakan untuk mengukur seberapa banyak dari total positif yang dapat dikenali oleh model. Lalu akurasi digunakan untuk mengukur seberapa sering model membuat prediksi yang benar dari seluruh prediksi yang dibuat. Pada nilai 0 memiliki presisi (0.99) dan recall (0.99), sedangkan pada nilai 1 memiliki presisi (0.98) dan recall (0.97). Kemudian pada metrik presisi (0.983819) menunjukkan bahwa dari semua prediksi positif (pasien meninggal), 98.39% benar-benar meninggal. Ini berarti model cukup baik dalam menahan false positives (meminimalkan kesalahan prediksi pasien meninggal). Pada metrik recall (0.968153) menunjukkan bahwa dari semua pasien yang benar-benar meninggal, model hanya dapat mengidentifikasi 96.82% dari mereka. Ini mengindikasikan model masih sering mengalami false negatives (salah memprediksi pasien yang meninggal sebagai tidak meninggal). Terakhir akurasi (0.985000) yang berarti akurasi keseluruhan model adalah 98.5% dari semua prediksi model sesuai dengan label yang sebenarnya

## NAIVE BAYES

### a. Import Library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

**Gambar 1.** Syntax import library

Melakukan import library pandas dan numpy untuk mengelola dan memanipulasi data juga untuk melakukan operasi pada array dan matriks multidimensi, sedangkan mengimpor library matplotlib dan seaborn untuk membuat berbagai jenis plot untuk visualisasi data. Kemudian mengimpor beberapa library dari scikit-learn yang menyediakan berbagai alat dan fungsi penting untuk membangun model machine learning, diantaranya ada LabelEncoder, StandardScaler, train\_test\_split, KNeighborsClassifier, classification\_report, confusion\_matrix, dan ConfusionMatrixDisplay.

### b. Pemanggilan data

```
# Pemanggilan data
datafarah= pd.read_csv('/content/heart_failure_clinical_records.csv')
datafarah
```

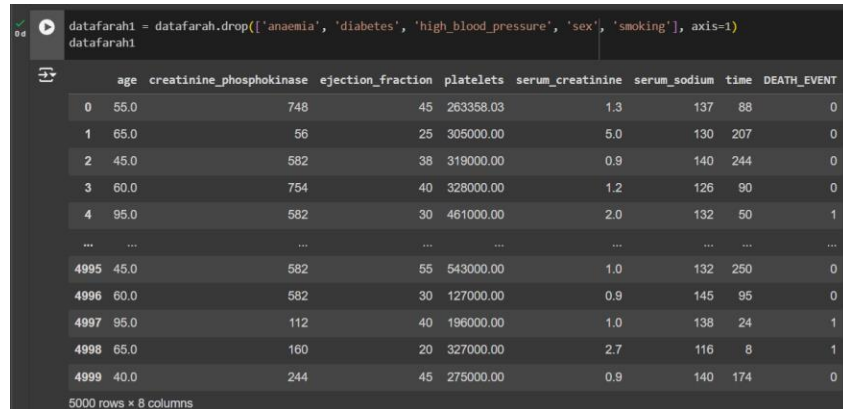
	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	55.0	0	748	0	45	0	263358.03	1.3	137	1	1	88	0
1	65.0	0	56	0	25	0	306000.00	5.0	130	1	0	207	0
2	45.0	0	582	1	38	0	319000.00	0.9	140	0	0	244	0
3	60.0	1	754	1	40	1	328000.00	1.2	126	1	0	90	0
4	95.0	1	582	0	30	0	461000.00	2.0	132	1	0	50	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	45.0	0	582	1	55	0	543000.00	1.0	132	0	0	290	0
4996	60.0	1	582	0	30	1	127000.00	0.9	145	0	0	95	0
4997	95.0	1	112	0	40	1	196000.00	1.0	136	0	0	24	1
4998	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1
4999	40.0	0	244	0	45	1	275000.00	0.9	140	0	0	174	0

5000 rows x 13 columns

**Gambar 2.** Syntax dan output memanggil data

Memanggil data dengan format csv dengan membuat nama data adalah datafarah dan menghasilkan output data sebanyak 5000 baris dan 13 kolom.

c. Imputasi data



The screenshot shows a Jupyter Notebook cell with the following code and output:

```
datafarah1 = datafarah.drop(['anaemia', 'diabetes', 'high_blood_pressure', 'sex', 'smoking'], axis=1)
datafarah1
```

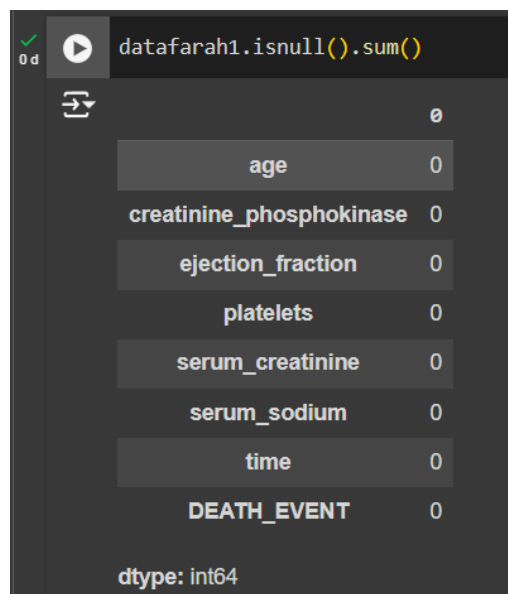
	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium	time	DEATH_EVENT
0	55.0	748	45	263358.03	1.3	137	88	0
1	65.0	56	25	305000.00	5.0	130	207	0
2	45.0	582	38	319000.00	0.9	140	244	0
3	60.0	754	40	328000.00	1.2	126	90	0
4	95.0	582	30	461000.00	2.0	132	50	1
...	...	...	...	...	...	...	...	...
4995	45.0	582	55	543000.00	1.0	132	250	0
4996	60.0	582	30	127000.00	0.9	145	95	0
4997	95.0	112	40	196000.00	1.0	138	24	1
4998	65.0	160	20	327000.00	2.7	116	8	1
4999	40.0	244	45	275000.00	0.9	140	174	0

5000 rows x 8 columns

**Gambar 3.** Syntax dan output imputasi data

Menggunakan fungsi drop dan axis=1 untuk menghapus variabel data berdasarkan kolom. Dapat dilihat ada beberapa variabel yang dihapus yaitu 'aneamia', 'diabetes', 'high\_blood\_pressure', 'sex', dan 'smoking'.

d. Data kosong



The screenshot shows a Jupyter Notebook cell with the following code and output:

```
datafarah1.isnull().sum()
```

	0
age	0
creatinine_phosphokinase	0
ejection_fraction	0
platelets	0
serum_creatinine	0
serum_sodium	0
time	0
DEATH_EVENT	0

dtype: int64

**Gambar 4.** Syntax dan output melihat data kosong

Berdasarkan output di atas dapat diketahui bahwa tidak ada variabel yang memiliki data kosong.



e. Duplikasi data

```
[5] datafarah1.duplicated().sum()
3739

[7] datafarah1.drop_duplicates(inplace=True)
```

**Gambar 5.** Syntax dan output melihat dan menghapus duplikasi data

Melihat duplikasi data menggunakan fungsi duplicated dan menghasilkan output duplikasi data sebanyak 3739 data, kemudian dengan fungsi drop\_duplicates untuk menghapus data yang duplikat.

f. Informasi data

```
datafarah1.info()

<class 'pandas.core.frame.DataFrame'>
Index: 1261 entries, 0 to 4972
Data columns (total 8 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                ---
0   age                                    1261 non-null   float64
1   creatinine_phosphokinase              1261 non-null   int64
2   ejection_fraction                    1261 non-null   int64
3   platelets                             1261 non-null   float64
4   serum_creatinine                      1261 non-null   float64
5   serum_sodium                          1261 non-null   int64
6   time                                  1261 non-null   int64
7   DEATH_EVENT                           1261 non-null   int64
dtypes: float64(3), int64(5)
memory usage: 88.7 KB
```

**Gambar 6.** Syntax dan output melihat informasi data

Berdasarkan output di atas dapat diketahui bahwa setelah data duplikasi di hapus, kini data memiliki 7 kolom variabel dan sebanyak 1261 baris dengan tipe data nya ada float dan integer.

g. Statistika deskriptif data

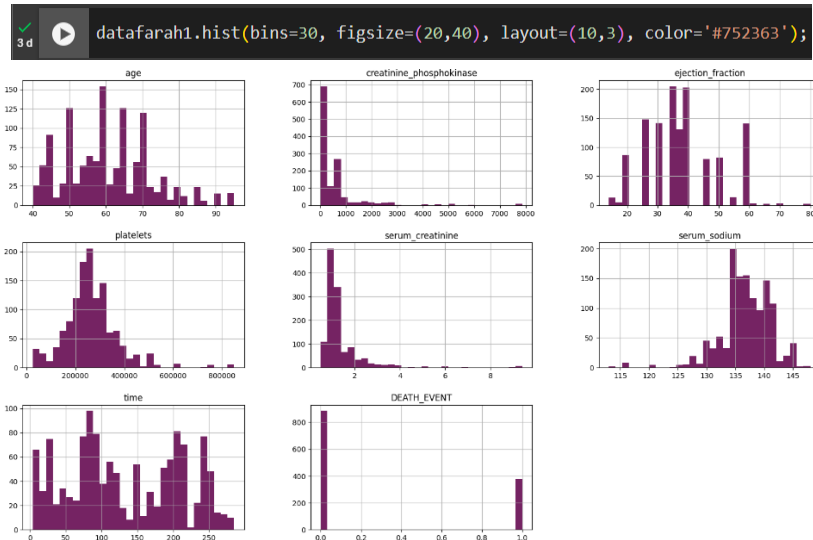
```
datafarah1.describe()
```

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium	time	DEATH_EVENT
count	1261.000000	1261.000000	1261.000000	1261.000000	1261.000000	1261.000000	1261.000000	1261.000000
mean	60.521283	581.321967	37.958763	263926.970880	1.365527	136.659001	132.246630	0.298176
std	11.811132	989.096533	11.682637	106171.161848	1.017499	4.398897	77.689861	0.457639
min	40.000000	23.000000	14.000000	25100.000000	0.500000	113.000000	4.000000	0.000000
25%	50.000000	113.000000	30.000000	208000.000000	0.900000	134.000000	74.000000	0.000000
50%	60.000000	246.000000	38.000000	263358.030000	1.100000	137.000000	117.000000	0.000000
75%	69.000000	582.000000	45.000000	310000.000000	1.300000	140.000000	205.000000	1.000000
max	95.000000	7861.000000	80.000000	850000.000000	9.400000	148.000000	285.000000	1.000000

**Gambar 7.** Syntax dan output statistika deskriptif data

Melihat statistika deskriptif dari data menggunakan fungsi describe dan menghasilkan output count, mean, standar deviasi, nilai minimum dan maksimum, kuartil atas, median, dan kuartil bawah untuk setiap variabel yang ada pada data.

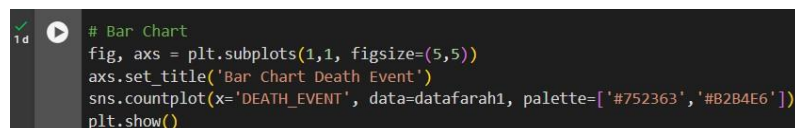
#### h. Histogram dari Dataset

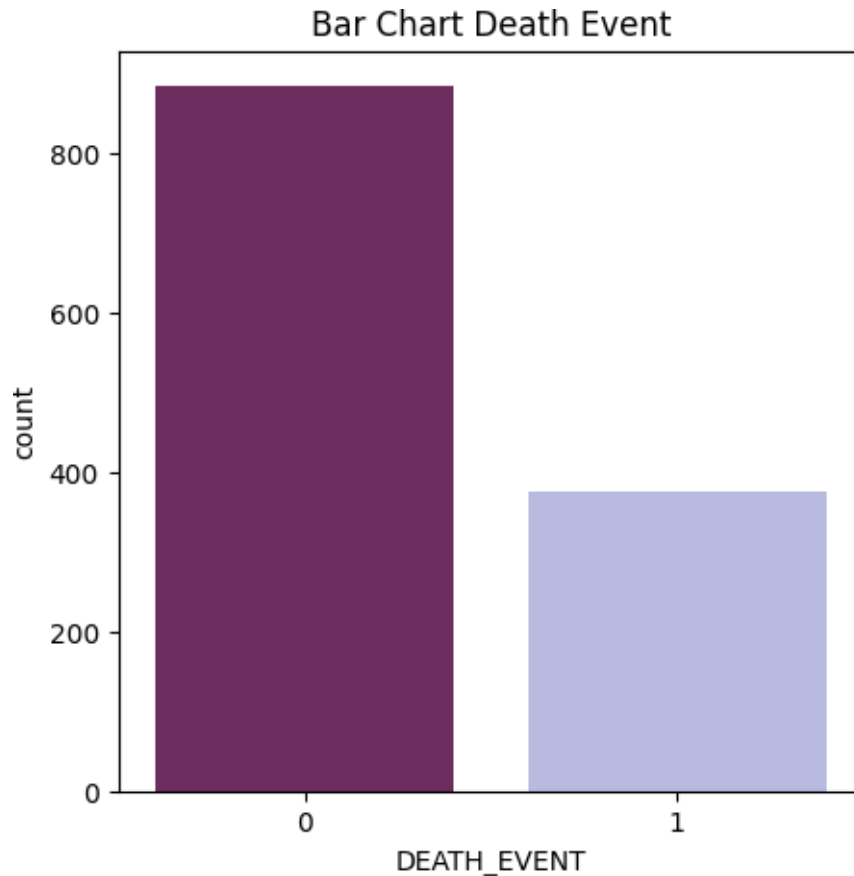


**Gambar 8.** Syntax dan output Histogram dari Dataset

Plot di atas menampilkan distribusi histogram dari berbagai variabel, pada distribusi usia terlihat yang mayoritas terdiri dari pasien berusia 50-70 tahun. Serum Creatinine dan Creatinine Phosphokinase menunjukkan nilai yang sebagian besar terkonsentrasi di level rendah dengan beberapa outlier tinggi, yang mungkin menandakan kondisi kritis. Distribusi Serum Sodium terpusat di kisaran normal (130-140), sementara Platelets (trombosit) kebanyakan berada di antara 150.000-300.000. Ejection Fraction (fraksi ejeksi) umumnya berada pada 30-40, dengan sedikit pasien memiliki nilai tinggi. Waktu tindak lanjut atau kelangsungan hidup tersebar merata, dan histogram Death Event (kematian) menunjukkan mayoritas pasien masih hidup, namun ada sejumlah yang meninggal. Data ini memberikan wawasan yang dapat membantu dalam analisis lebih lanjut terkait faktor-faktor medis yang memengaruhi kelangsungan hidup pasien.

#### i. Bar Chart dari Death Event



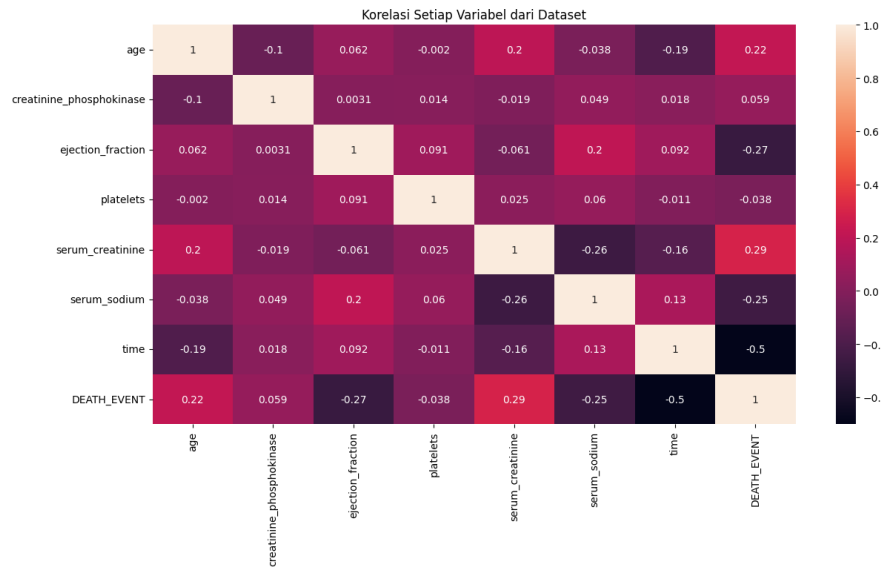


**Gambar 9.** Syntax dan output Bar Chart dari Death Event

Bar chart di atas menggambarkan distribusi kejadian kematian pada pasien, dengan dua kategori: pasien yang bertahan hidup (nilai 0) dan yang meninggal (nilai 1). Sebagian besar pasien dalam dataset ini berada di kategori bertahan hidup, dengan jumlah lebih dari 800 pasien. Sebaliknya, jumlah pasien yang meninggal jauh lebih sedikit, sekitar 400. Hal ini menunjukkan bahwa tingkat kelangsungan hidup pasien lebih tinggi dibandingkan dengan tingkat kematian, berdasarkan data yang ada.

j. Heatmap Correlation dari Dataset

```
1 d  ▶ farah_correlation = datafarah1.corr(method='pearson')  
  
plt.figure(figsize=(14,7))  
plt.title("Korelasi Setiap Variabel dari Dataset")  
sns.heatmap(data=farah_correlation, annot=True)  
plt.show()
```



**Gambar 10.** Syntax dan output Heatmap dari Dataset

Heatmap di atas menunjukkan korelasi antar variabel dalam dataset. Beberapa hubungan penting yang terlihat adalah: usia memiliki korelasi positif dengan kejadian kematian (0.22) dan kadar kreatinin serum (0.2), menunjukkan semakin tua usia pasien, semakin tinggi risiko kematian dan gangguan fungsi ginjal. Fraksi ejeksi memiliki korelasi negatif dengan kematian (-0.27), artinya semakin rendah fraksi ejeksi, semakin tinggi risiko kematian. Kadar kreatinin serum juga berkorelasi positif dengan kematian (0.29), mengindikasikan bahwa gangguan fungsi ginjal meningkatkan risiko kematian. Waktu tindak lanjut memiliki korelasi negatif kuat dengan kematian (-0.5), menunjukkan bahwa pasien yang bertahan lebih lama memiliki risiko kematian lebih rendah. Korelasi antara variabel lainnya relatif lemah.

k. Membagi dan mengubah data

```
[14] X = datafarah1.iloc[:, :-1].values
      y = datafarah1.iloc[:, 7].values

[15] X_train, X_test, y_train, y_test = train_test_split(X, y,
      test_size=0.1, shuffle=True, stratify=y, random_state=42)

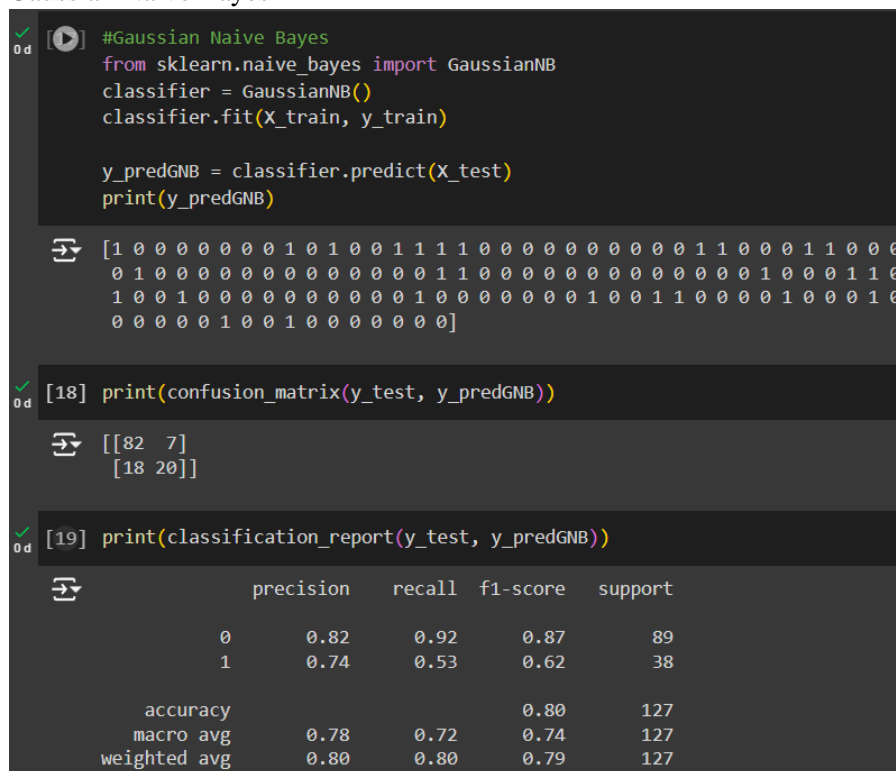
[16] print("Jumlah baris X_train:", len(X_train))
      print("Jumlah baris X_test:", len(X_test))
      print("Jumlah baris y_train:", len(y_train))
      print("Jumlah baris y_test:", len(y_test))

Jumlah baris X_train: 1134
Jumlah baris X_test: 127
Jumlah baris y_train: 1134
Jumlah baris y_test: 127
```

**Gambar 11.** Syntax dan output membagi dan mengubah data

Output di atas menunjukkan proses pembagian data menjadi data pelatihan (train) dan data pengujian (test) menggunakan fungsi `train_test_split`. Pertama, dilakukan persiapan data dengan variabel `x` dan `y` menggunakan fungsi `iloc`. Kedua, pembagian data dengan fungsi `train_test_split` yang digunakan untuk membagi data `x` dan `y` menjadi data pelatihan dan data pengujian, fungsi `test_size=0.1` berarti 10% dari data digunakan sebagai data pengujian, dan sisanya 90% digunakan sebagai data pelatihan, fungsi `shuffle=True` memastikan data diacak sebelum dibagi, fungsi `stratify=y` memastikan bahwa proporsi kelas yang sama di `y` tetap dipertahankan dalam data pelatihan dan pengujian, fungsi `random_state=42` digunakan untuk memastikan hasil yang dapat direproduksi. Ketiga, setelah pemisahan data didapatkan `x_train` dan `y_train` berisi data pelatihan sedangkan `x_test` dan `y_test` berisi data pengujian. Kemudian `x_train` dan `y_train` memiliki 1134 baris juga `x_test` dan `y_test` memiliki 127 baris.

#### 1. Uji Gaussian Naive Bayes



```
#Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_predGNB = classifier.predict(X_test)
print(y_predGNB)
```

```
[1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0
 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0
 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0]
```

```
[18] print(confusion_matrix(y_test, y_predGNB))
```

```
[[82  7]
 [18 20]]
```

```
[19] print(classification_report(y_test, y_predGNB))
```

	precision	recall	f1-score	support
0	0.82	0.92	0.87	89
1	0.74	0.53	0.62	38
accuracy			0.80	127
macro avg	0.78	0.72	0.74	127
weighted avg	0.80	0.80	0.79	127

**Gambar 12.** Syntax dan output uji Gaussian Naive Bayes

Model Gaussian Naive Bayes pada dataset ini memiliki akurasi keseluruhan 79%, dengan kinerja yang lebih baik dalam memprediksi kelas 0 dibandingkan kelas 1. Hal ini terlihat dari nilai recall untuk kelas 0 yang mencapai 92%, menunjukkan bahwa sebagian besar instance kelas 0 berhasil diprediksi dengan benar, sementara untuk kelas 1 hanya 53%. Precision untuk kelas 0 juga lebih tinggi (82%) dibandingkan kelas 1 (74%), yang berarti model lebih sering benar ketika memprediksi kelas 0. Namun, model agak kesulitan dalam mengidentifikasi kelas 1 dengan benar, terlihat dari nilai f1-score yang lebih rendah (0.62)

### m. Uji Multinomial Naive Bayes

[illegible]

Multinomial Naive Bayes pada dataset ini menunjukkan performa yang cukup baik dalam mengenali kelas 0 dengan precision 0.76 dan recall 0.87, artinya model dapat dengan baik memprediksi sebagian besar sampel kelas 0 dengan tingkat kesalahan yang relatif rendah. Namun, model ini mengalami kesulitan dalam mengenali kelas 1, dengan precision hanya 0.54 dan recall 0.37, yang menunjukkan bahwa banyak sampel kelas 1 yang salah diklasifikasikan sebagai kelas 0. Akurasi keseluruhan model ini adalah 72%, dengan f1-score yang lebih baik pada kelas 0 (0.81) dibandingkan kelas 1 (0.44), menandakan ketidakseimbangan performa antara kedua kelas. Berdasarkan output matriks kebingungan dapat diketahui pada prediksi 0 (true positive) sebanyak 77 kali model memprediksi dengan benar bahwa pasien tidak meninggal dan prediksi 0 (false positive) sebanyak 24 kali model salah memprediksi bahwa pasien tidak meninggal, padahal sebenarnya meninggal. Kemudian pada prediksi 1 (true negative) sebanyak 14 kali model memprediksi dengan benar bahwa

pasien meninggal dan prediksi 1 (false negative) sebanyak 12 kali model salah memprediksi bahwa pasien meninggal, padahal sebenarnya tidak.

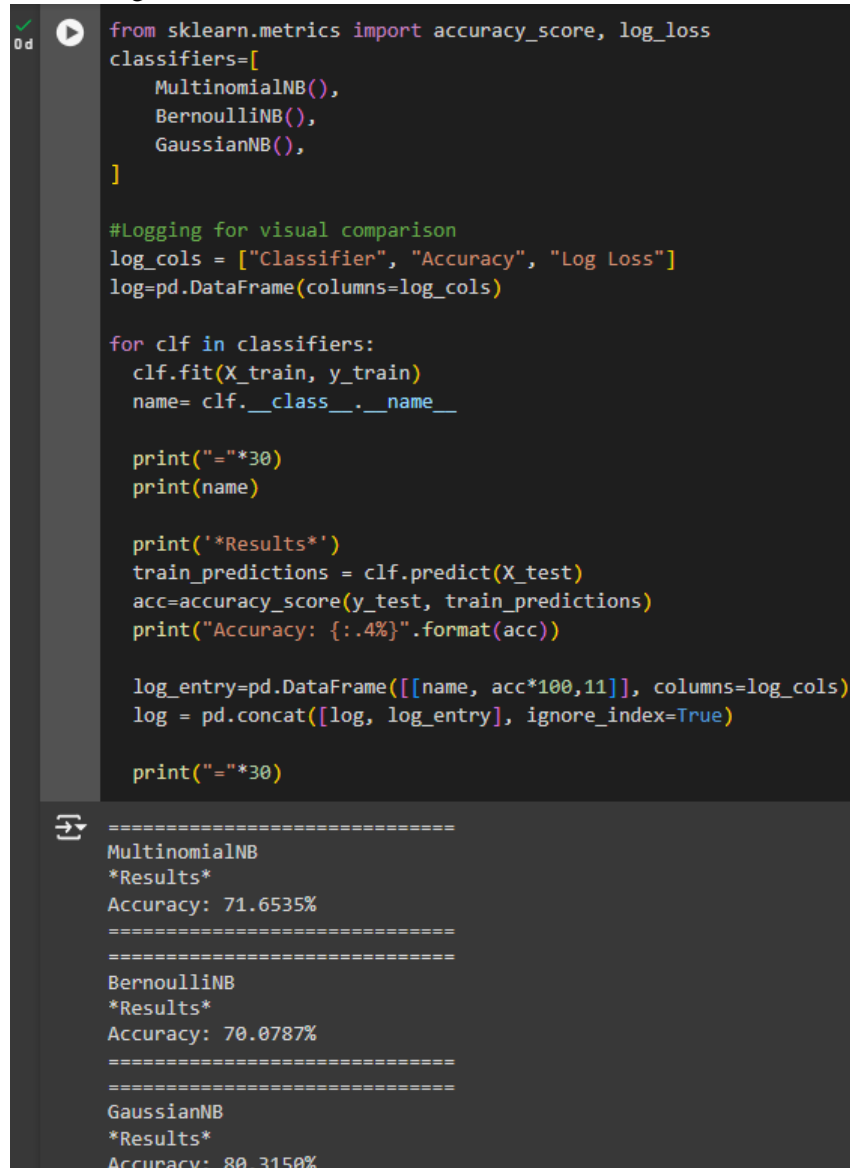
n. Uji Bernouli Naive Bayes

[illegible]

**Gambar 14.** Syntax dan output uji Bernouli Naive Bayes

Pada Bernoulli Naive Bayes, model menunjukkan performa yang sangat bias terhadap kelas 0. Dari matriks kebingungan, semua sampel kelas 0 (89) diklasifikasikan dengan benar (recall 1.00), sementara tidak ada satu pun sampel kelas 1 yang terdeteksi dengan benar (recall 0.00), sehingga semua sampel kelas 1 (38) salah diklasifikasikan sebagai kelas 0. Precision untuk kelas 1 adalah 0.00 karena model tidak mampu mengidentifikasi sampel kelas 1 sama sekali. Secara keseluruhan, akurasi model 70% didominasi oleh kemampuan mengenali kelas 0, namun performanya sangat buruk dalam mendeteksi kelas 1, sehingga model ini tidak cocok untuk situasi di mana kedua kelas harus dikenali dengan baik.

o. Akurasi dari ketiga klasifikasi



```
from sklearn.metrics import accuracy_score, log_loss
classifiers=[
    MultinomialNB(),
    BernoulliNB(),
    GaussianNB(),
]

#Logging for visual comparison
log_cols = ["Classifier", "Accuracy", "Log Loss"]
log=pd.DataFrame(columns=log_cols)

for clf in classifiers:
    clf.fit(X_train, y_train)
    name= clf.__class__.__name__

    print("="*30)
    print(name)

    print('*Results*')
    train_predictions = clf.predict(X_test)
    acc=accuracy_score(y_test, train_predictions)
    print("Accuracy: {:.4%}".format(acc))

    log_entry=pd.DataFrame([[name, acc*100,11]], columns=log_cols)
    log = pd.concat([log, log_entry], ignore_index=True)

    print("="*30)
```

=====

MultinomialNB

\*Results\*

Accuracy: 71.6535%

=====

BernoulliNB

\*Results\*

Accuracy: 70.0787%

=====

GaussianNB

\*Results\*

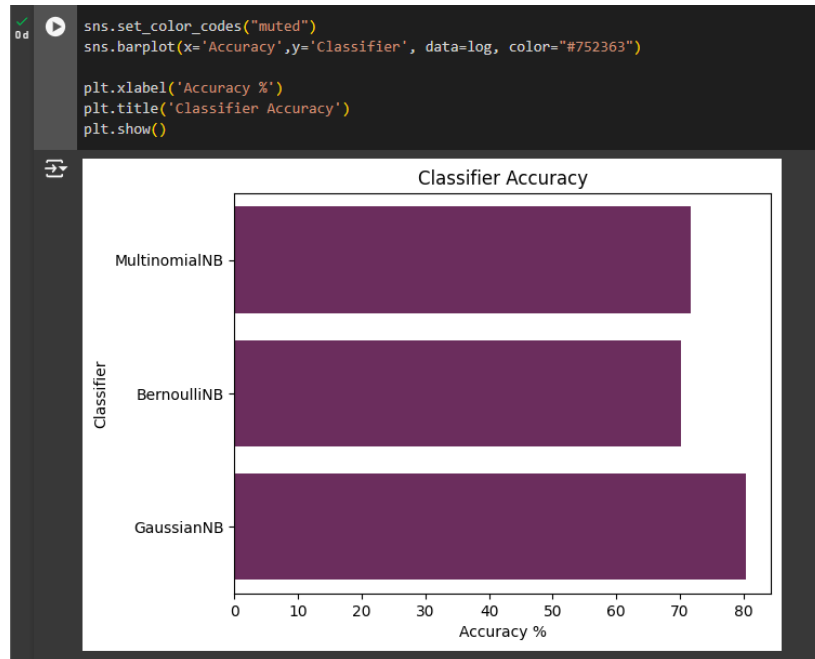
Accuracy: 80.3150%

**Gambar 15.** Syntax dan output akurasi dari ketiga klasifikasi

Berdasarkan output diketahui bahwa ketiga model Naive Bayes menghasilkan akurasi yang berbeda pada dataset yang sama. GaussianNB memiliki akurasi tertinggi sebesar 80.32%, menunjukkan bahwa model ini paling baik dalam memprediksi kelas dengan benar dibandingkan dua model lainnya. MultinomialNB mengikuti dengan akurasi 71.65%, yang cukup baik namun masih lebih rendah dari GaussianNB. Sedangkan BernoulliNB memiliki akurasi terendah sebesar 70.08%, menunjukkan bahwa pendekatan ini kurang efektif pada dataset ini dibandingkan dengan yang lain. Secara keseluruhan, model GaussianNB lebih unggul dalam hal performa akurasi



p. Bar Plot dari Akurasi



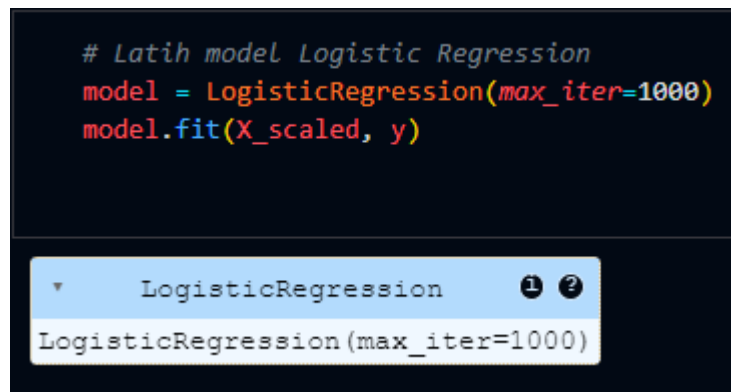
**Gambar 12.** Syntax dan output Bar Plot akurasi

Bar plot yang ditampilkan menggambarkan perbandingan akurasi dari tiga model Naive Bayes: MultinomialNB, BernoulliNB, dan GaussianNB. Dari grafik ini terlihat bahwa GaussianNB memiliki akurasi tertinggi, mencapai hampir 80%, menunjukkan bahwa model ini paling baik dalam memprediksi hasil yang benar. MultinomialNB memiliki akurasi sekitar 71%, sedikit lebih baik dari BernoulliNB, yang akurasinya mendekati 70%.

## LOGISTIC REGRESSION

### a. Melatih model Logistic Regression

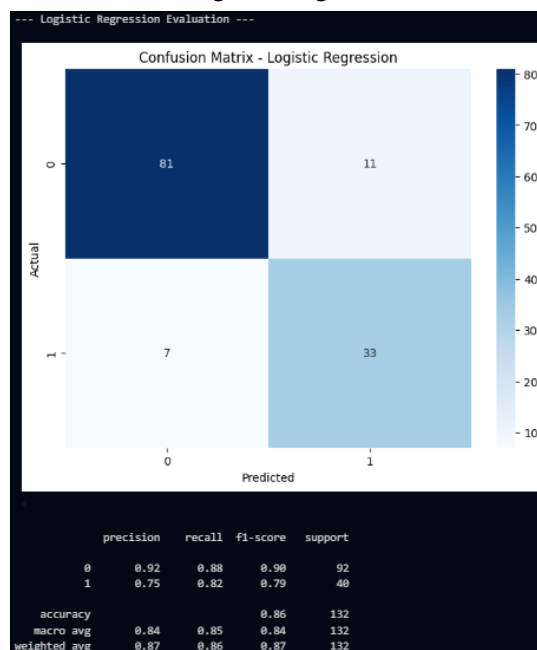
```
# Latih model Logistic Regression
model = LogisticRegression(max_iter=1000)
model.fit(X_scaled, y)
```



**Gambar 1.** Melatih model Logistic Regression

model `LogisticRegression` diinisialisasi dengan parameter `max_iter=1000`, yang berfungsi untuk menentukan batas maksimum iterasi saat proses pelatihan agar model dapat mencapai konvergensi. Hal ini penting terutama jika dataset yang digunakan cukup kompleks atau jumlah fitur yang besar. Setelah model dibuat, dilakukan pelatihan dengan memanggil fungsi `fit()` menggunakan data fitur `X_scaled` dan label target `y`. Data `X_scaled` menunjukkan bahwa fitur-fitur telah melalui proses normalisasi sebelumnya. Selain itu, tampilan antarmuka menunjukkan pratinjau objek model `LogisticRegression(max_iter=1000)`, menandakan bahwa model telah berhasil diinisialisasi dengan parameter yang ditentukan.

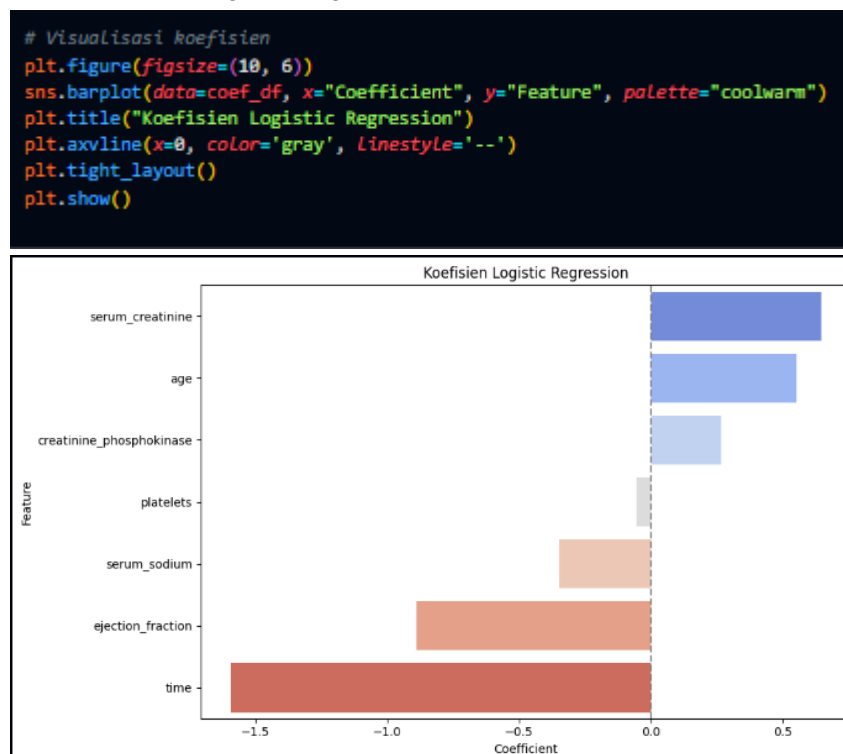
### b. Evaluasi Performa Logistic Regression Model



**Gambar 2.** Confusion Matrix dan Evaluasi Model Logistic Regression

Gambar ini menunjukkan hasil evaluasi model *Logistic Regression* yang terdiri dari dua bagian utama: *confusion matrix* dan metrik klasifikasi. Confusion matrix menggambarkan jumlah prediksi benar dan salah yang dilakukan oleh model. Dari matriks tersebut terlihat bahwa, Sebanyak 81 data kelas 0 berhasil diprediksi dengan benar sebagai kelas 0 (*true negative*). Sebanyak 11 data kelas 0 salah diprediksi sebagai kelas 1 (*false positive*). Sebanyak 33 data kelas 1 diprediksi dengan benar sebagai kelas 1 (*true positive*). Sebanyak 7 data kelas 1 salah diprediksi sebagai kelas 0 (*false negative*). Di bawah confusion matrix, ditampilkan hasil evaluasi model menggunakan metrik klasifikasi: precision, recall, f1-score, dan support. Untuk kelas 0, precision adalah 0.92, recall 0.88, dan f1-score 0.90 dari total 92 sampel. Untuk kelas 1, precision adalah 0.75, recall 0.82, dan f1-score 0.79 dari total 40 sampel. Secara keseluruhan, akurasi model mencapai 86%, dengan rata-rata makro (macro avg) dari f1-score sebesar 0.84 dan rata-rata berbobot (weighted avg) sebesar 0.87. Ini menunjukkan bahwa model *Logistic Regression* memiliki performa yang cukup baik, terutama pada kelas mayoritas (kelas 0), namun masih bisa ditingkatkan lagi untuk kelas minoritas (kelas 1).

### c. Visualisasi Koefisien Logistic Regression



**Gambar 3.** Syntax dan output Bar Plot Visualisasi Koefisien

Bar plot ini tampilan menggambarkan nilai koefisien dari masing-masing fitur pada model Logistic Regression. Grafik ini menunjukkan seberapa besar pengaruh setiap fitur terhadap

prediksi model. Fitur `time` dan `ejection_fraction` memiliki koefisien negatif terbesar, yang berarti semakin besar nilainya, semakin kecil kemungkinan prediksi model terhadap kelas positif. Sebaliknya, fitur seperti `serum_creatinine` dan `age` memiliki koefisien positif yang cukup besar, menunjukkan bahwa peningkatan nilai pada fitur-fitur ini cenderung meningkatkan peluang prediksi ke kelas positif. Visualisasi ini membantu dalam interpretasi model, karena memudahkan untuk memahami fitur mana yang paling berpengaruh dalam pengambilan keputusan oleh model Logistic Regression.

## Decision Tree

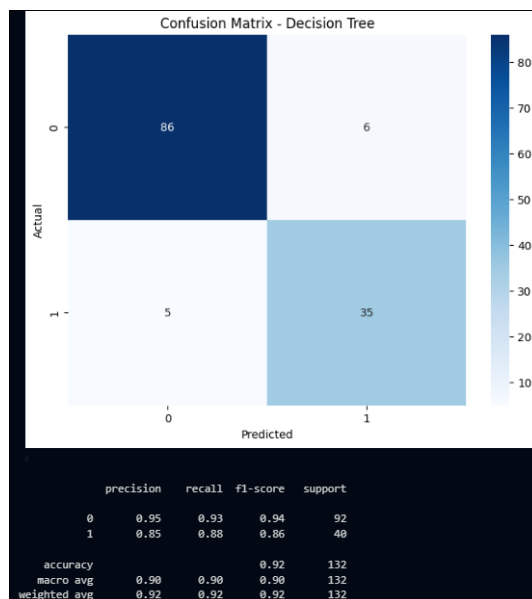
### a. Melatih model Decision Tree

```
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
y_prob_dt = dt_model.predict_proba(X_test)[:, 1] # untuk kurva ROC
```

**Gambar 1.** pelatihan model Decision Tree

proses pelatihan model Decision Tree Classifier menggunakan dataset pelatihan ( $X_{\text{train}}$ ,  $y_{\text{train}}$ ) serta prediksi terhadap data uji ( $X_{\text{test}}$ ). Model dibuat dengan `random_state=42` untuk memastikan hasil yang konsisten setiap kali dijalankan. Setelah model dilatih (`fit`), dilakukan prediksi kelas (`predict`) dan prediksi probabilitas (`predict_proba`) untuk data uji. Nilai probabilitas dari kelas positif (`[:, 1]`) digunakan untuk perhitungan kurva ROC yang berguna dalam evaluasi kinerja klasifikasi model berbasis probabilitas.

### b. Evaluasi model Decision Tree

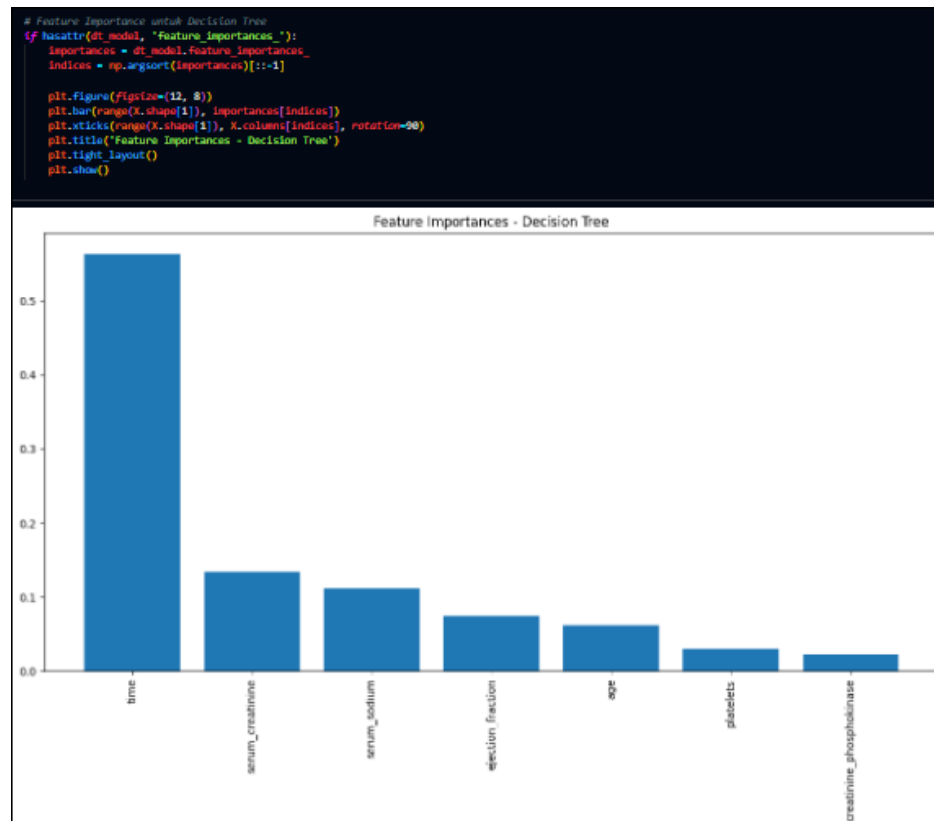


**Gambar 2.** Confusion matrix

Confusion matrix dan metrik evaluasi model Decision Tree Classifier. Dari confusion matrix, diketahui bahwa model memprediksi kelas 0 dengan benar sebanyak 86 kali dan salah sebanyak 6 kali (False Positive). Sementara itu, untuk kelas 1, model memprediksi benar sebanyak 35 kali dan salah 5 kali (False Negative). Dari segi metrik evaluasi, Precision untuk kelas 0 adalah 0.95, menunjukkan bahwa dari semua prediksi kelas 0, 95% adalah benar. Untuk kelas 1, precision-nya 0.85. Recall untuk kelas 0 adalah 0.93, artinya model

berhasil menangkap 93% dari seluruh data aktual kelas 0, sedangkan recall untuk kelas 1 adalah 0.88. F1-score, yang merupakan rata-rata harmonik dari precision dan recall, bernilai 0.94 untuk kelas 0 dan 0.86 untuk kelas 1. Akurasi keseluruhan model adalah 92%, menunjukkan bahwa 92% prediksi model sesuai dengan label yang sebenarnya dari total 132 data uji.

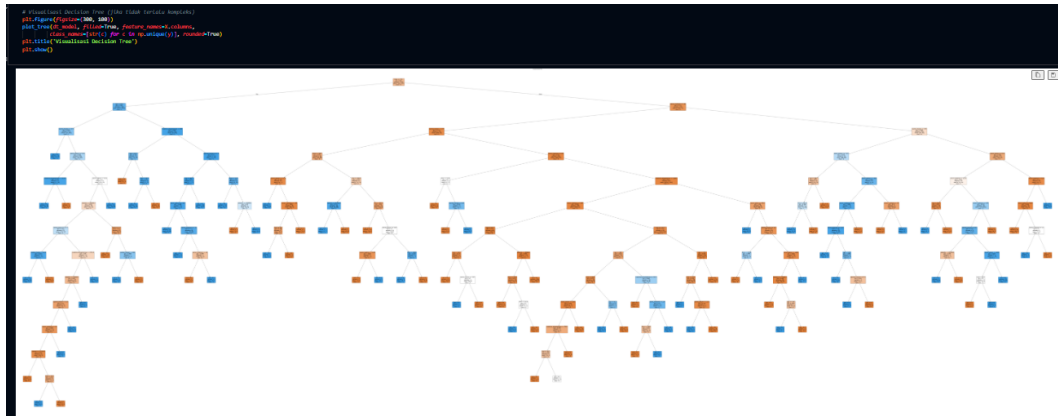
### c. Feature Importance Decision Tree



**Gambar 3.** Syntax dan Output Bar Plot Feature Importance Decision Tree

Bar plot yang ditampilkan menunjukkan pentingnya masing-masing fitur dalam model Decision Tree. Dari grafik ini terlihat bahwa fitur 'time' memiliki kontribusi paling besar terhadap keputusan model, dengan nilai feature importance tertinggi, yaitu lebih dari 0.5. Ini menunjukkan bahwa atribut 'time' sangat berpengaruh dalam proses klasifikasi atau prediksi. Fitur-fitur lain seperti 'serum\_creatinine', 'serum\_sodium', dan 'ejection\_fraction' juga memiliki pengaruh, meskipun jauh lebih kecil dibandingkan 'time'. Sedangkan fitur seperti 'creatinine\_phosphokinase' dan 'platelets' memberikan kontribusi paling kecil dalam model, yang berarti pengaruhnya terhadap output prediksi relatif rendah.

#### d. Visualisasi Decision Tree



**Gambar 4.** Syntax dan Output Visualisasi Decision Tree

visualisasi lengkap dari struktur model Decision Tree yang telah dilatih. Setiap node dalam pohon mewakili keputusan berdasarkan nilai fitur tertentu, sementara cabang menunjukkan hasil dari keputusan tersebut. Warna pada node menggambarkan kelas mayoritas (biru untuk satu kelas dan oranye untuk kelas lainnya), dan tingkat kecerahan menunjukkan tingkat kepastian (semakin terang berarti lebih tidak pasti). Meskipun visualisasi ini memberikan wawasan menyeluruh tentang bagaimana model membuat prediksi, namun karena kedalaman dan kompleksitas pohon yang tinggi, visual ini bisa menjadi sulit untuk dianalisis secara manual. Visualisasi ini sangat berguna dalam memahami alur logika model, mendeteksi kemungkinan overfitting, dan mengevaluasi fitur mana yang sering digunakan untuk pengambilan keputusan.

#### Kesimpulan

- Model yang dipilih memiliki akurasi tertinggi di antara keempat model yang diuji.
- Namun, akurasi bukan satu-satunya indikator Precision, Recall, dan F1 Score juga perlu dipertimbangkan terutama jika data tidak seimbang.
- Logistic Regression cocok jika interpretabilitas penting, sedangkan Decision Tree memberikan visualisasi pohon keputusan yang intuitif.
- KNN bergantung pada pemilihan nilai  $K$ , dan performanya dapat menurun jika data sangat besar.
- Naive Bayes bekerja baik pada data yang bersifat *independen antar fitur*.

Pemilihan akhir model tetap harus mempertimbangkan konteks aplikasi, kebutuhan interpretasi, serta performa menyeluruh.