

PRAKTIKUM DATA WAREHOUSING DAN DATA MINING
MODUL 9
ALGORITMA DECISION TREE (POHON KEPUTUSAN)



Disusun oleh:
Adinda Aulia Hapsari
L200220037

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA
TAHUN 2024

Setelah kegiatan selesai, lembar kerja ini dicetak (di-print) dan dikumpulkan ke asisten.	(Diisi oleh Asisten)
NIM : L200220037	Nilai Praktek :
Nama : Adinda Aulia Hapsari	
Nama Asisten : Diva Halimah	Tanda Tangan :
Tanggal Praktikum : 29 November 2024	

KEGIATAN PRAKTIKUM

Pada kegiatan praktikum ini, kita menggunakan pohon keputusan untuk membuat klasifikasi pada data iris. Dengan metode ini, suatu data uji dapat diklasifikasikan berdasarkan kelas datanya.

Langkah-langkah menggunakan algoritma decision tree dengan bahasa pemrograman python adalah sebagai berikut:

1. Bukalah jupyter notebook yang sudah terinstal pada komputer yang digunakan.
2. Buatlah file baru pada jupyter notebook dengan mengklik menu New-Python 3, secara otomatis akan keluar window baru untuk melakukan coding pada jupyter nootebook. Pastikan file yang dibuat dengan dataset yang digunakan berapa dalam satu folder.
3. Pada baris pertama tuliskan beberapa library yang akan digunakan dalam praktikum. Ada beberapa library seperti numpy, pandas, matplotlib, seaborn dan lain sebagainya.

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

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

4. Setelah itu lakukan read dataset menggunakan library pandas, disini kita menggunakan dataset iris dengan ekstensi .csv. Kita akan melakukan beberapa hal untuk mengeksplorasi data yang digunakan. Pertama kita akan melihat isi dataset yang dipakai menggunakan fungsi head, fungsi tersebut akan menampilkan isi data beserta atribut yang ada. Terdapat 6 atribut yang digunakan pada dataset tersebut yaitu Id, SepalLengthCm, SepalWidthCm, PetalLengthCM, PetalWidthCm dan Species yang nanti digunakan sebagai label klasifikasi.

```
[2]: iris=pd.read_csv('iris.csv')
[3]: iris.head(5)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

5. Kita juga bisa mengecek bentuk dari data yang digunakan menggunakan fungsi shape, bisa juga melihat value apa saja yang ada pada atribut Species.

```
[4]: #cek jumlah baris dan kolom
iris.shape

[4]: (150, 6)

[5]: iris['Species'].unique()

[5]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

6. Hal lain yang harus kita lakukan adalah memeriksa apakah dataset memiliki nilai kosong (null) atau tidak, hal ini bisa dilakukan dengan menggunakan fungsi info(). Terlihat tidak ada data yang kosong pada semua atribut, ini menunjukkan dataset yang digunakan sangat baik.

```
[6]: iris.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ---
 0   Id                 150 non-null   int64
 1   SepalLengthCm      150 non-null   float64
 2   SepalWidthCm       150 non-null   float64
 3   PetalLengthCm      150 non-null   float64
 4   PetalWidthCm       150 non-null   float64
 5   Species            150 non-null   object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

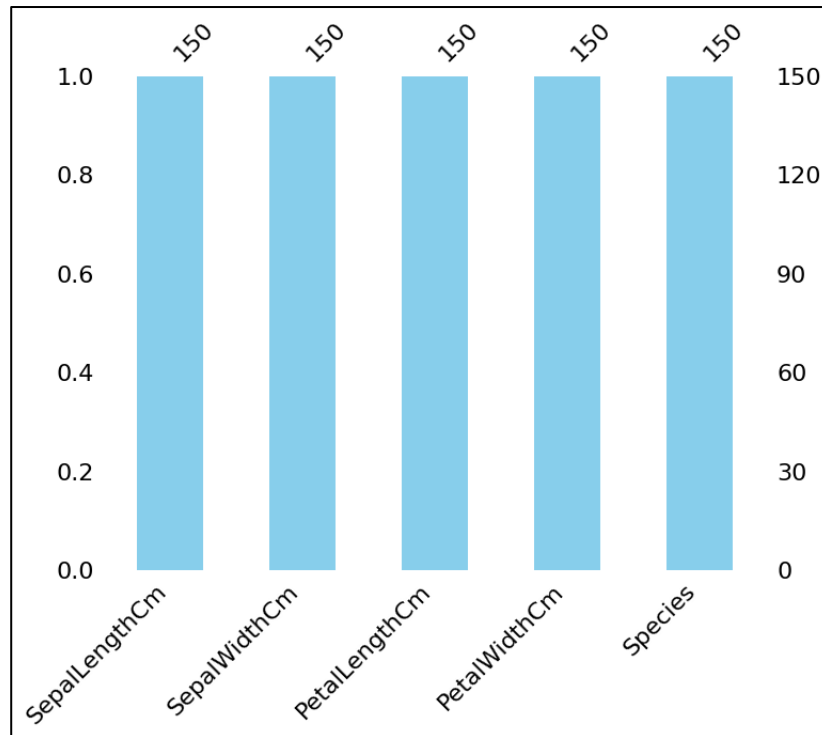
7. Dari semua atribut yang ada kita akan memakai lima fitur saja yaitu SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm dan Species. Sehingga atribut Id akan dihapus karena fitur tersebut tidak memiliki pengaruh dalam proses klasifikasi, penghapusan atribut bisa menggunakan fungsi drop.

```
[7]: iris.drop(columns="Id",inplace=True)
iris.isnull().sum()

[7]: SepalLengthCm    0
     SepalWidthCm    0
     PetalLengthCm    0
     PetalWidthCm    0
     Species         0
     dtype: int64
```

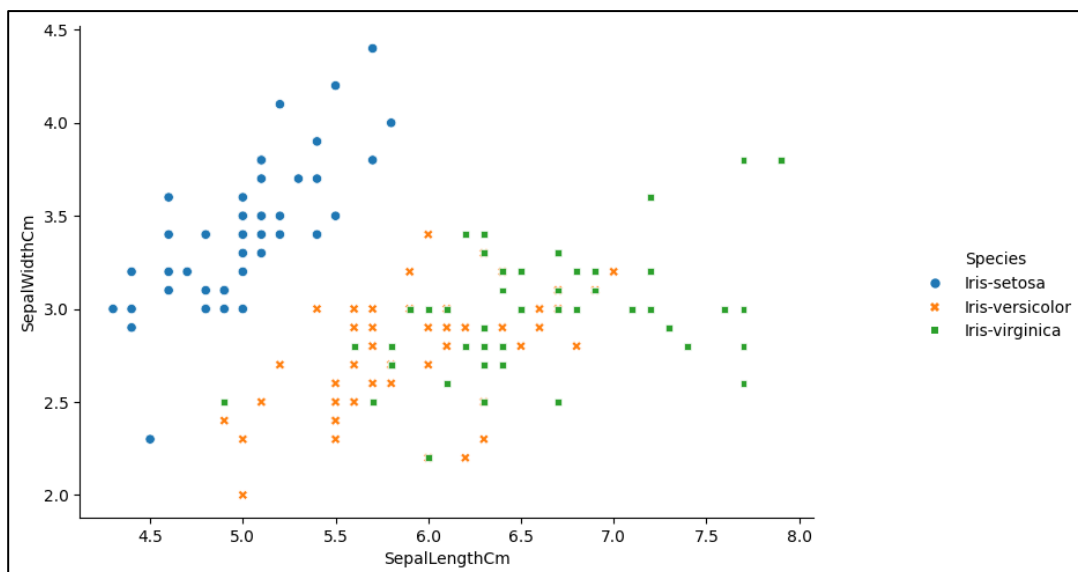
8. Kita juga bisa mengecek data yang kosong menggunakan grafik.

```
[8]: import missingno as msno
     msno.bar(iris,figsize=(8,6),color='skyblue')
     plt.show()
```



9. Proses selanjutnya adalah melakukan visualisasi data, proses ini sangat penting dilakukan untuk mengetahui sebaran data yang ada dalam dataset.

```
[10]: #scatterplot
g=sns.relplot(x='SepalLengthCm',y='SepalWidthCm',data=iris,hue='Species',style='Species')
g.fig.set_size_inches(10,5)
plt.show()
```



10. Kemudian kita bisa mencari korelasi pada setiap fitur yang digunakan untuk melakukan klasifikasi.

```
[33]: iris_numeric = iris.select_dtypes(include='number')
iris_corr = iris_numeric.corr()
print(iris_corr)
```

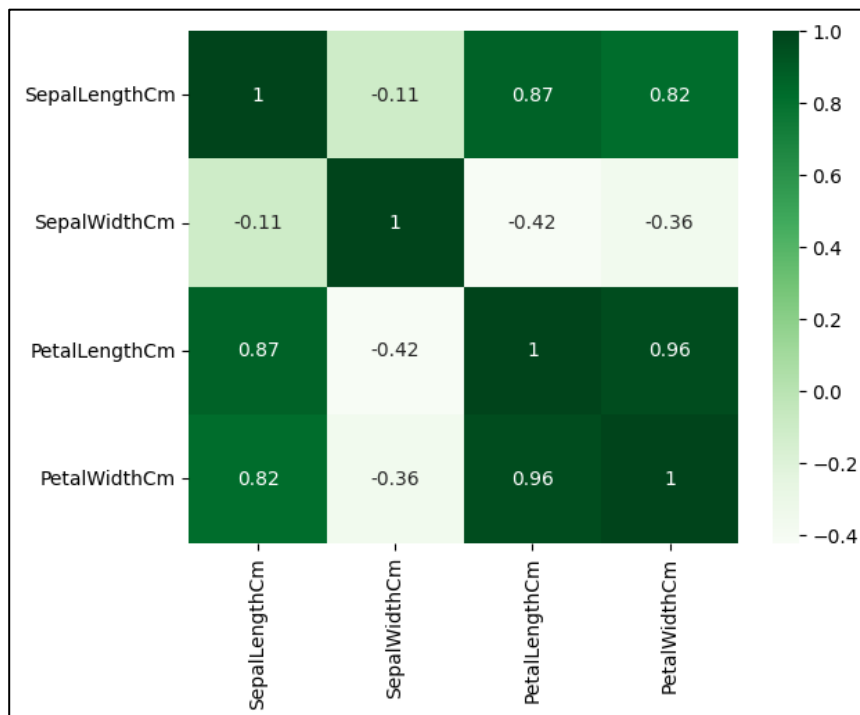
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

```
[34]: # Pilih hanya kolom numerik
iris_numeric = iris.select_dtypes(include='number')

# Hitung korelasi
iris_corr = iris_numeric.corr()

# Buat heatmap
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(iris_corr, annot=True, cmap='Greens')
plt.show()
```



- Hal penting lainnya yang harus dilakukan adalah memisahkan antara fitur dan label, atribut Species akan digunakan sebagai label dan atribut lainnya digunakan sebagai fitur pada proses klasifikasi.

Data Preprocessing

```
[35]: #pisahkan antara fitur dan label
x = iris.drop('Species', axis = 1)
y = iris['Species']
```

12. Kita bisa mengecek atribut yang sudah terpisah dengan menampilkan variabel X yang merupakan fitur dan variabel y sebagai label.

```
[36]: x
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
[37]: y
```

0	Iris-setosa
1	Iris-setosa
2	Iris-setosa
3	Iris-setosa
4	Iris-setosa
...	...
145	Iris-virginica
146	Iris-virginica
147	Iris-virginica
148	Iris-virginica
149	Iris-virginica

Name: Species, Length: 150, dtype: object

13. Pada dataset iris, label yang digunakan bukan merupakan data numerical, padahal mesin hanya bisa memahami angka, sehingga kita harus merubah isi dari label menjadi angka, proses ini dinamakan dengan encoding.

```
[38]: #label encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
y
```

```
[38]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
         2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
         2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

14. Sebelum memasuki proses klasifikasi data perlu dibagi menjadi dua bagian yaitu data training dan data testing. Data training akan digunakan untuk pembuatan model, sedangkan data testing digunakan untuk melakukan evaluasi pada model yang sudah dibuat. proses pembagian dataset ini menggunakan library sklearn. Terlihat pada gambar dataset terbagi menjadi 120 untuk data training dan 30 untuk data testing.

```
[39]: #splitting data menjadi train data dan test data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=0)

print('The shape of x_train is: {}'.format(x_train.shape))
print('The shape of x_test is: {}'.format(x_test.shape))
print('The shape of y_train is: {}'.format(y_train.shape))
print('The shape of y_test is: {}'.format(y_test.shape))
```

```
The shape of x_train is: (120, 4)
The shape of x_test is: (30, 4)
The shape of y_train is: (120,)
The shape of y_test is: (30,)
```

15. Tahap selanjutnya adalah pembuatan model menggunakan algoritma Naive Bayes. Kita bisa memanggilnya dari library sklearn dan menaruhnya pada variabel model.

Membuat Model

```
[40]: #model training
model = DecisionTreeClassifier()
model.fit(x_train,y_train)
```

```
[40]: DecisionTreeClassifier
```

```
DecisionTreeClassifier()
```

16. Kita lakukan training model menggunakan algoritma naive bayes pada data training. Disini kita mendapatkan akurasi pada saat training memperoleh skor 100%.

```
[41]: #prediksi pada data train
pred_train = model.predict(x_train)

cm = confusion_matrix(y_train, pred_train)

#confusion matrix
print('Confusion matrix Decision Tree\n',cm)
print('')

#akurasi
print('Akurasi pada saat training: {}'.format(accuracy_score(y_train,pred_train))) #confusion matrix

Confusion matrix Decision Tree
[[39  0  0]
 [ 0 37  0]
 [ 0  0 44]]

Akurasi pada saat training: 1.0
```

17. Langkah terakhir adalah menguji model yang dibuat dengan data testing yang sudah disiapkan. Pada saat evaluasi model kita mendapatkan nilai yang sama pada akurasi, presisi, recall dan f1 score, yaitu sebesar 100%.

```
[42]: #pred pada data test
pred_test = model.predict(x_test)

[43]: cm = confusion_matrix(y_test, pred_test)
accuracy = accuracy_score(y_test, pred_test)
precision = precision_score(y_test, pred_test, average='micro')
recall = recall_score(y_test, pred_test, average='micro')
f1 = f1_score(y_test, pred_test, average='micro')

print('Confusion matrix for DecisionTree\n',cm)
print('')
print('Akurasi pada data test: %.3f' %accuracy)
print('precision: %.3f' %precision)
print('recall: %.3f' %recall)
print('f1-score: %.3f' %f1)

Confusion matrix for DecisionTree
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]

Akurasi pada data test: 1.000
precision: 1.000
recall: 1.000
f1-score: 1.000
```


TUGAS

1. Buatlah eksplorasi data dari dataset heart_failure.csv!

```
[44]: hf=pd.read_csv('heart_failure.csv')
```

```
[47]: hf.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype  
---  -
 0   age                        299 non-null   float64
 1   anaemia                   299 non-null   int64  
 2   creatinine_phosphokinase  299 non-null   int64  
 3   diabetes                  299 non-null   int64  
 4   ejection_fraction        299 non-null   int64  
 5   high_blood_pressure       299 non-null   int64  
 6   platelets                 299 non-null   float64
 7   serum_creatinine          299 non-null   float64
 8   serum_sodium              299 non-null   int64  
 9   sex                       299 non-null   int64  
10  smoking                   299 non-null   int64  
11  time                      299 non-null   int64  
12  DEATH_EVENT               299 non-null   int64  
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

2. Cari 5 fitur yang memiliki pengaruh paling besar terhadap proses klasifikasi!

```
[51]: #hitung korelasi
correlation_matrix = hf.corr()
print(correlation_matrix['DEATH_EVENT'].sort_values(ascending=False))

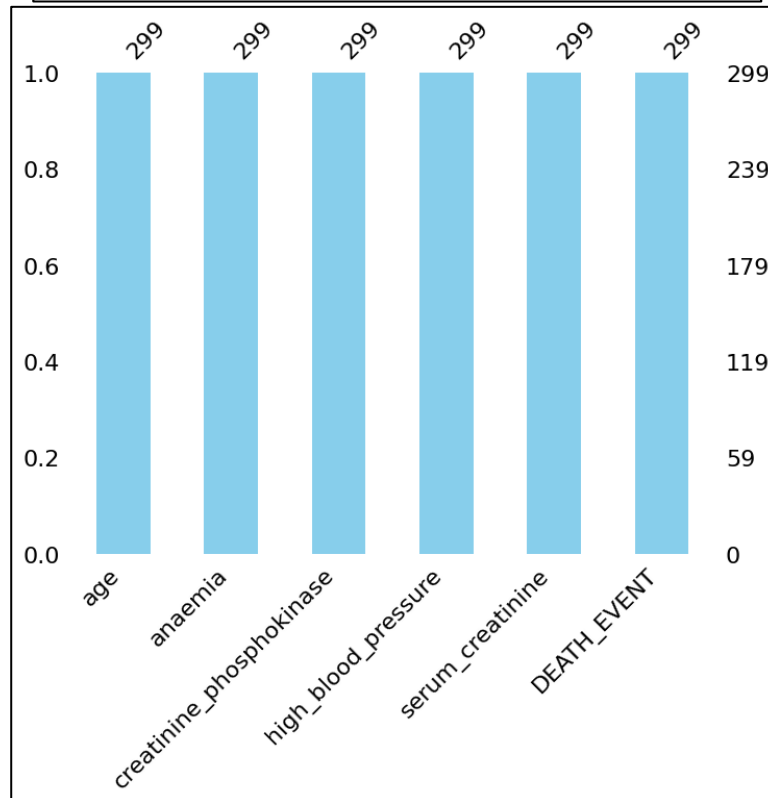
DEATH_EVENT      1.000000
serum_creatinine  0.294278
age              0.253729
high_blood_pressure  0.079351
anaemia          0.066270
creatinine_phosphokinase  0.062728
diabetes         -0.001943
sex              -0.004316
smoking          -0.012623
platelets        -0.049139
serum_sodium     -0.195204
ejection_fraction -0.268603
time             -0.526964
Name: DEATH_EVENT, dtype: float64
```

```
[53]: hf.drop(columns=["diabetes", "sex", "smoking", "platelets", "serum_sodium", "ejection_fraction", "time"], inplace=True)
hf.isnull().sum()
```

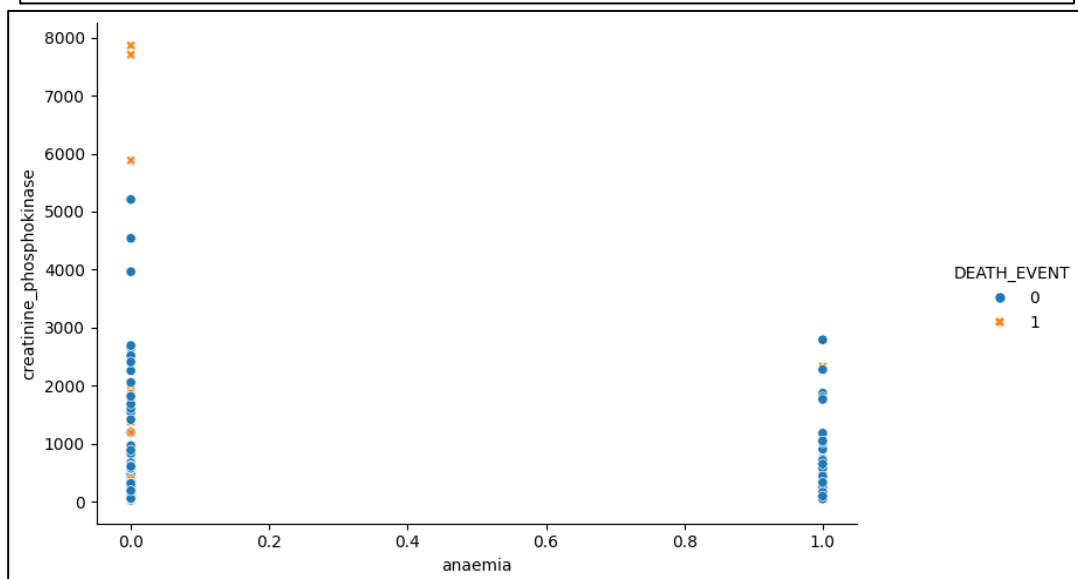
```
[53]: age              0
anaemia            0
creatinine_phosphokinase  0
high_blood_pressure  0
serum_creatinine   0
DEATH_EVENT        0
dtype: int64
```

3. Gunakan 5 fitur yang sudah dipilih untuk melakukan klasifikasi!

```
[54]: import missingno as msno
      msno.bar(hf, figsize=(8,6), color='skyblue')
      plt.show()
```



```
[56]: #scatterplot
      g=sns.relplot(x='anaemia',y='creatinine_phosphokinase',data=hf,hue='DEATH_EVENT',style='DEATH_EVENT')
      g.fig.set_size_inches(10,5)
      plt.show()
```



```
[57]: hf_numeric = hf.select_dtypes(include='number')
hf_corr = hf_numeric.corr()
print(hf_corr)
```

	age	anaemia	creatinine_phosphokinase	\
age	1.000000	0.088006	-0.081584	
anaemia	0.088006	1.000000	-0.190741	
creatinine_phosphokinase	-0.081584	-0.190741	1.000000	
high_blood_pressure	0.093289	0.038182	-0.070590	
serum_creatinine	0.159187	0.052174	-0.016408	
DEATH_EVENT	0.253729	0.066270	0.062728	

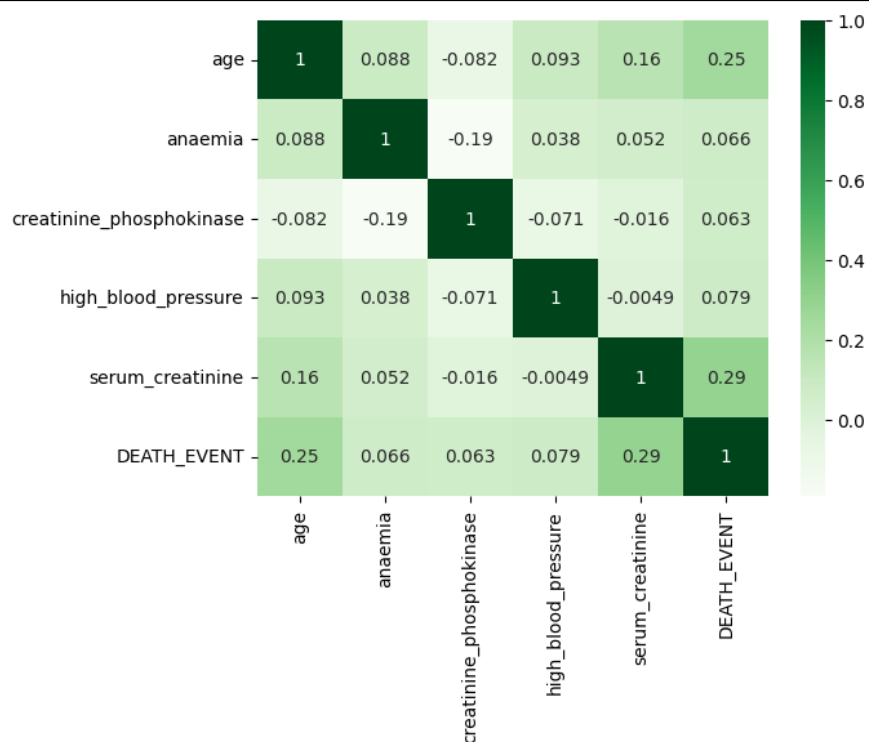
	high_blood_pressure	serum_creatinine	DEATH_EVENT
age	0.093289	0.159187	0.253729
anaemia	0.038182	0.052174	0.066270
creatinine_phosphokinase	-0.070590	-0.016408	0.062728
high_blood_pressure	1.000000	-0.004935	0.079351
serum_creatinine	-0.004935	1.000000	0.294278
DEATH_EVENT	0.079351	0.294278	1.000000

```
[58]: # Pilih hanya kolom numerik
hf_numeric = hf.select_dtypes(include='number')

# Hitung korelasi
hf_corr = hf_numeric.corr()

# Buat heatmap
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(hf_corr, annot=True, cmap='Greens')
plt.show()
```



```
[59]: #pisahkan antara fitur dan label
x = hf.drop('DEATH_EVENT', axis = 1)
y = hf['DEATH_EVENT']
```

```
[60]: x
```

	age	anaemia	creatinine_phosphokinase	high_blood_pressure	serum_creatinine
0	75.0	0	582	1	1.9
1	55.0	0	7861	0	1.1
2	65.0	0	146	0	1.3
3	50.0	1	111	0	1.9
4	65.0	1	160	0	2.7
...
294	62.0	0	61	1	1.1
295	55.0	0	1820	0	1.2
296	45.0	0	2060	0	0.8
297	45.0	0	2413	0	1.4
298	50.0	0	196	0	1.6

299 rows × 5 columns

```
[61]: y
```

0	1
1	1
2	1
3	1
4	1
..	
294	0
295	0
296	0
297	0
298	0

Name: DEATH_EVENT, Length: 299, dtype: int64

```
[62]: #label encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
y
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

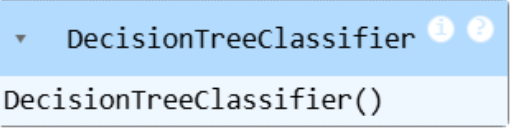
```
[63]: #splitting data menjadi train data dan test data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

print('The shape of x_train is: {}'.format(x_train.shape))
print('The shape of x_test is: {}'.format(x_test.shape))
print('The shape of y_train is: {}'.format(y_train.shape))
print('The shape of y_test is: {}'.format(y_test.shape))

The shape of x_train is: (239, 5)
The shape of x_test is: (60, 5)
The shape of y_train is: (239,)
The shape of y_test is: (60,)
```

4. Buatlah model untuk algoritma decision tree dan hitunglah nilai akurasi, presisi, recall dan F1-score!

```
[64]: #model training
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
```

[64]:  DecisionTreeClassifier()

```
[65]: #prediksi pada data train
pred_train = model.predict(x_train)

cm = confusion_matrix(y_train, pred_train)

#confusion matrix
print('Confusion matrix Decision Tree\n', cm)
print('')

#akurasi
print('Akurasi pada saat training: {}'.format(accuracy_score(y_train, pred_train))) #confusion matrix

Confusion matrix Decision Tree
[[166  0]
 [ 1 72]]

Akurasi pada saat training: 0.99581589958159
```

```
[66]: #pred pada data test
pred_test = model.predict(x_test)
```

```
[67]: cm = confusion_matrix(y_test, pred_test)
accuracy = accuracy_score(y_test, pred_test)
precision = precision_score(y_test, pred_test, average='micro')
recall = recall_score(y_test, pred_test, average='micro')
f1 = f1_score(y_test, pred_test, average='micro')

print('Confusion matrix for DecisionTree\n', cm)
print('')
print('Akurasi pada data test: %.3f' % accuracy)
print('precision: %.3f' % precision)
print('recall: %.3f' % recall)
print('f1-score: %.3f' % f1)

Confusion matrix for DecisionTree
[[32  5]
 [12 11]]

Akurasi pada data test: 0.717
precision: 0.717
recall: 0.717
f1-score: 0.717
```