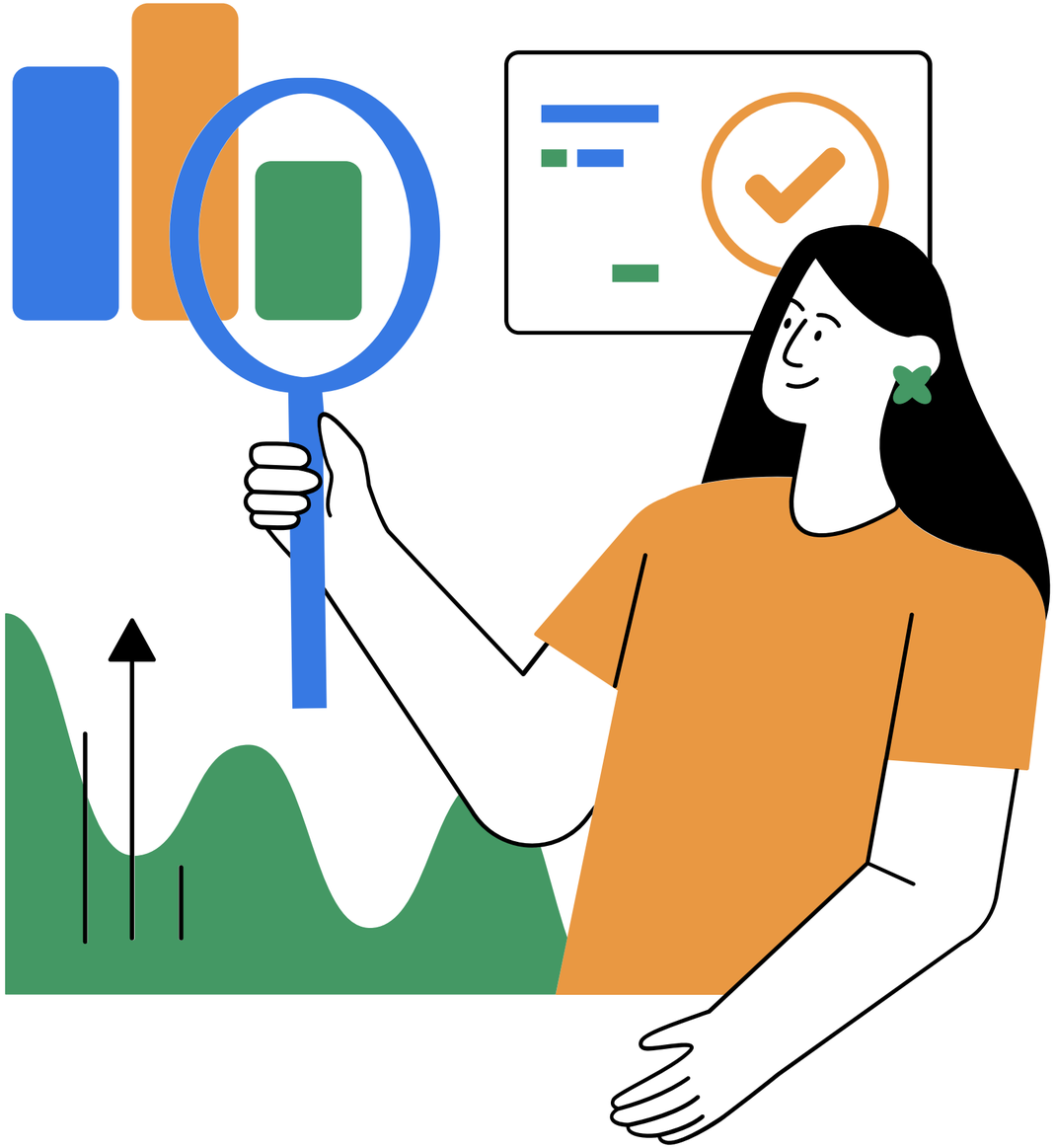
Kelompok 4

Analisis Heart Failure Clinical Records

With KNN & Cross Validation

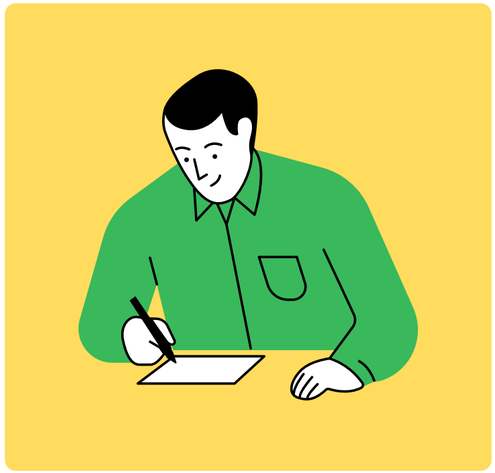
Dosen Pengampu: Endang Yuliani, S.Mat., M.Si



28 NOVEMBER 2025

Anggota Kelompok

Rafi King Akbar
1314623018



Hansen Juliantino
1314623044

Oki Ramadhan Pramono
1314623067

Kelompok 4

Daftar Isi

Pendahuluan

Latar Belakang

Penyakit gagal jantung (*heart failure*) merupakan salah satu penyebab kematian utama di dunia. Dataset *Heart Failure Clinical Records* berisi 299 data pasien dengan berbagai parameter klinis seperti usia, tekanan darah, *ejection fraction*, serum *creatinine*, dan waktu perawatan. Data ini dapat dimanfaatkan untuk menganalisis faktor-faktor yang berpengaruh terhadap kematian serta membangun model prediksi awal risiko DEATH_EVENT. **Dengan menggunakan algoritma machine learning seperti K-Nearest Neighbors (KNN), analisis ini bertujuan membantu memberikan gambaran awal kondisi pasien berdasarkan pola data klinis yang tersedia.**

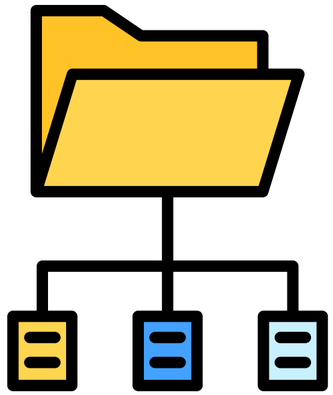
Tujuan Analisis

- **Membangun model prediksi** DEATH_EVENT menggunakan algoritma K-Nearest Neighbors (KNN)
- **Menentukan nilai K terbaik** melalui evaluasi 5 *Fold Cross Validation*
- **Menilai performa model** menggunakan metrik evaluasi seperti *accuracy*, *precision*, *recall*, *F1-score*.

Apa itu KNN?

KNN adalah algoritma klasifikasi sederhana yang memprediksi label sebuah data berdasarkan kedekatannya dengan data-data lain di dataset.

Semakin dekat suatu data dengan kelompok tertentu, semakin besar kemungkinan data tersebut memiliki label yang sama.



Bagaimana Cara Kerjanya?

1. **Hitung jarak data baru** dengan seluruh data pada dataset
2. **Ambil K tetangga terdekat**
3. **Voting Mayoritas**: label yang paling banyak muncul di antara tetangga yang akan dijadikan prediksi model

Sifat yang jarang dipunyai metode klasifikasi lain

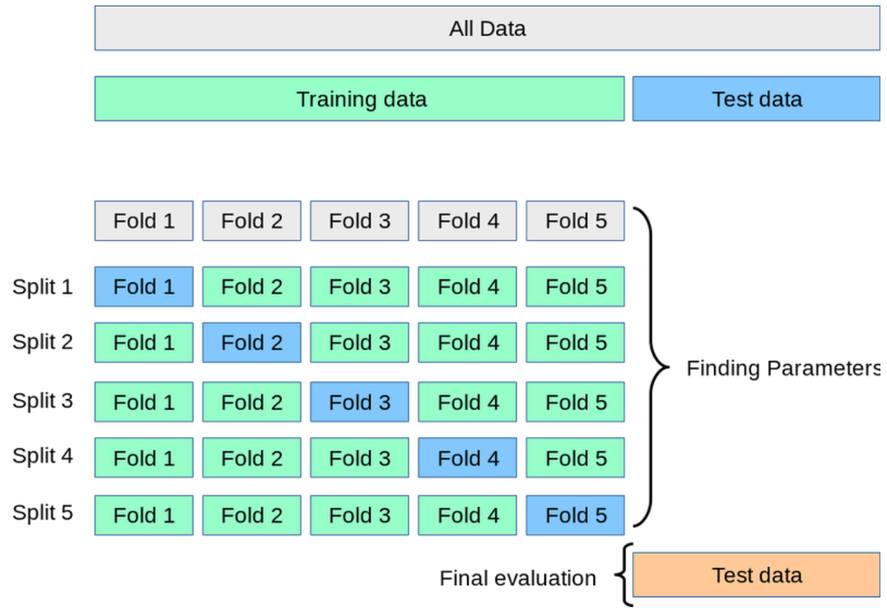
Bersifat **lazy learner**, tidak ada proses training model, hanya menghitung jarak saat prediksi.

Kelompok 4

Cross Validation

Cross Validation adalah teknik evaluasi model dengan cara membagi dataset menjadi beberapa bagian (fold), lalu melatih dan menguji model berkali-kali pada kombinasi data yang berbeda.

Tujuannya: mendapatkan estimasi performa yang lebih akurat dibanding train-test split biasa.



Bagaimana Pembagian Train Test dalam KFold?

Jika menggunakan K-Fold, maka setiap iterasi:

- Data Train = (K-1)/K * 100%
- Data Test = 1/K * 100%

Contoh 5 Fold:

- Data Train = 4/5 = 80%
- Data Test = 1/5 = 20%

Tahapan Pengolahan Data

Kelompok 4

1.Load Data

Data di-*load* dari sumber GitHub dan disiimpan dalam *DataFrame* Pandas

```
import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/rafikingakbar/KNNPlayground/refs/heads/main/data/heart_failure_clinical_records_dataset.xlsx%20-%20heart_failure_clinical
df.head()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75	0	582	0	20	1	265000	1,9	130	1	0	4	1
1	55	0	7861	0	38	0	263358,03	1,1	136	1	0	6	1
2	65	0	146	0	20	0	162000	1,3	129	1	1	7	1
3	50	1	111	0	20	0	210000	1,9	137	1	0	7	1
4	65	1	160	1	20	0	327000	2,7	116	0	0	8	1

Dari info di atas, diketahui terdapat **13 variabel** dalam dataset, mulai dari Age sampai dengan death_event

Kelompok 4

1.Load Data

Berikut ini fitur datanya

Variabel	Keterangan	Tipe Data
Age	Usia Pasien	Numerik
Anaemia	Status anemia pasien	Biner(0/1)
Creatinine_phospokinase	Kadar enzim CPK di dalam darah(mcg/L)	Numerik
Diabetes	Pasien menderita diabetes	Biner(0/1)
Ejection_fraction	Persentase darah yang keluar dari ventrikel kiri setiap kontraksi	Numerik
High_blood_pressure	Status tekanan darah tinggi pasien	Biner(0/1)
Platelets	Jumlah trombosit dalam darah(kiloplatelets/mL)	Numeri
Serum_creatinine	Kadar kreatine serum dalam darah(mg/dL)	Numerik
Serum_sodium	Kadar natrium serum dalam darah(mEq/L)	Numerik
Sex	Jenis kelamin pasien	Biner(0/1)
Smoking	Pasien merokok atau tidak	Biner(0/1)
Time	Periode tindak lanjut dari pasien(dalam hari)	Numerik
Death_event	Kejadian kematian dalam periode tindak lanjut	Biner(0/1)

2.Preprocessing

A. Melihat info data

Perintah **df.info()** menunjukkan bahwa dataset berisi 299 baris dan 13 kolom, semuanya tanpa missing value. Mayoritas kolom sudah bertipe int64, namun tiga kolom age, platelets, dan serum_creatinine masih bertipe object karena nilai desimal menggunakan koma. Oleh karena itu, kolom-kolom tersebut perlu dibersihkan dan dikonversi ke format numerik sebelum masuk ke tahap modelling.

```
df.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    object
 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    object
 7   serum_creatinine          299 non-null    object
 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: int64(10), object(3)
memory usage: 30.5+ KB
```

2.Preprocessing

B. Cek nilai unik untuk kolom Object

Pengecekan nilai unik menunjukkan bahwa kolom serum_creatinine, platelets, dan age masih terbaca sebagai object karena menggunakan koma pada nilai desimal. Format angka yang tidak seragam ini membuat Python menganggapnya sebagai string, sehingga perlu dilakukan pembersihan dan penyesuaian format agar bisa dikonversi menjadi numerik sebelum analisis lebih lanjut.

```
df["serum_creatinine"].unique()[:50]

array(['1,9', '1,1', '1,3', '2,7', '2,1', '1,2', '1,5', '9,4', '4', '0,9',
       '1', '0,8', '1,6', '1,83', '5,8', '3', '3,5', '2,3', '0,6', '4,4',
       '1,4', '6,8', '2,2', '2', '1,18', '2,9', '0,7', '1,7', '2,5',
       '1,8', '3,2', '0,75', '3,7', '3,4', '6,1', '2,4', '9', '5', '0,5',
       '3,8'], dtype=object)
```

```
df["platelets"].unique()[:50]

array(['265000', '263358,03', '162000', '210000', '327000', '204000',
       '127000', '454000', '388000', '368000', '253000', '136000',
       '276000', '427000', '47000', '262000', '166000', '237000', '87000',
       '297000', '289000', '149000', '196000', '284000', '153000',
       '200000', '360000', '319000', '302000', '188000', '228000',
       '226000', '321000', '305000', '329000', '185000', '218000',
       '194000', '310000', '271000', '451000', '140000', '395000',
       '418000', '351000', '255000', '461000', '223000', '216000',
       '254000'], dtype=object)
```

```
df["age"].unique()[:50]

array(['75', '55', '65', '50', '90', '60', '80', '62', '45', '49', '82',
       '87', '70', '48', '68', '53', '95', '58', '94', '85', '69', '72',
       '51', '57', '42', '41', '67', '79', '59', '44', '63', '86', '66',
       '43', '46', '61', '81', '52', '64', '40', '60,667', '73', '77',
       '78', '54', '47', '56'], dtype=object)
```

2.Preprocessing

C. Perbaikan format angka

Perbaikan format angka dilakukan dengan mengganti koma menjadi titik pada kolom serum_creatinine, platelets, dan age, karena nilai desimal yang bertanda koma membuatnya terbaca sebagai string.

Melalui *df[col].str.replace(",", ".")*, format angka diperbaiki sehingga kolom tersebut dapat dikonversi menjadi numerik dan siap digunakan untuk analisis maupun modelling.

```
clean_col = ["serum_creatinine", "platelets", "age"]

for col in clean_col:
    df[col] = df[col].str.replace(",", ".", regex=False)
```

2.Preprocessing

D. Mengubah kolom obj menjadi num

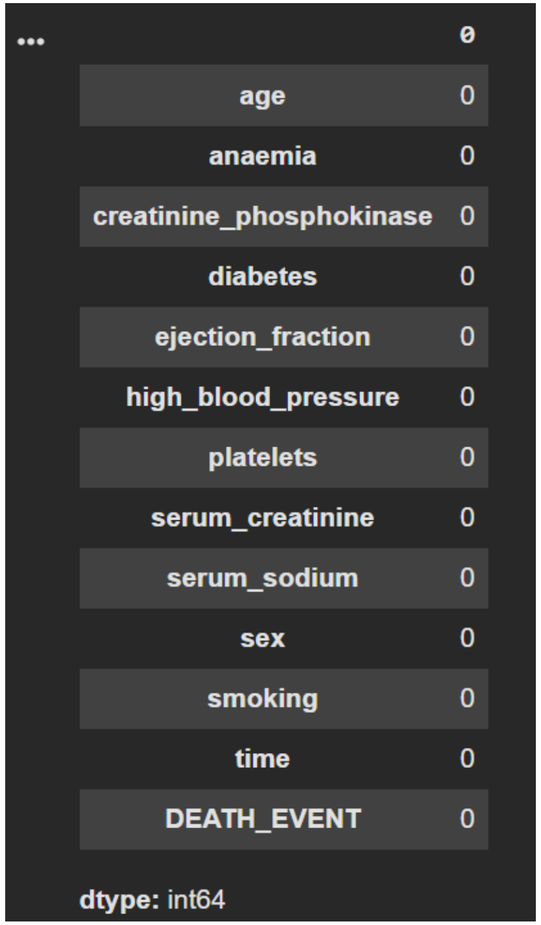
Setelah format angka diperbaiki, kolom serum_creatinine, platelets, dan age dikonversi dari object menjadi numerik menggunakan pd.to_numeric(). Parameter **errors="coerce"** digunakan agar nilai yang tidak dapat dikonversi otomatis menjadi NaN, sehingga proses analisis dan modelling dapat berjalan tanpa error. Langkah ini penting karena model hanya dapat memproses data dalam format numerik.

```
convert_col = ["serum_creatinine", "platelets", "age"]

for col in convert_col:
    df[col] = pd.to_numeric(df[col], errors="coerce")
```

2.Preprocessing

E. Cek Missing Values

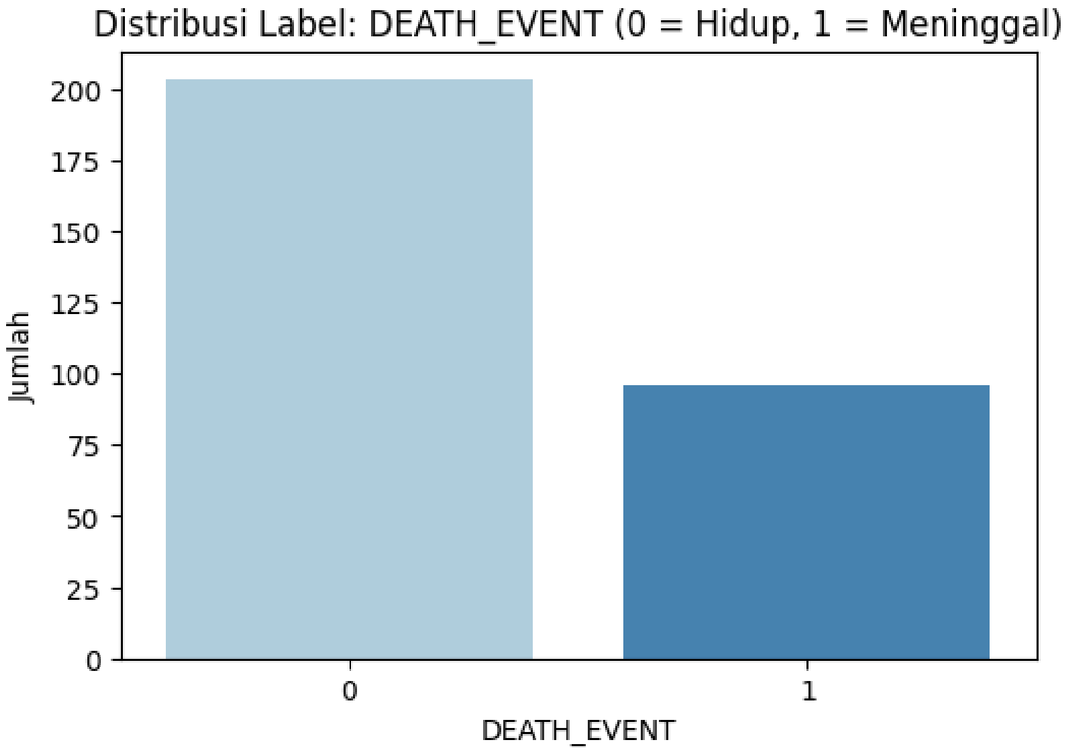
Hasil **df.isnull().sum()** menunjukkan seluruh kolom memiliki nilai 0, sehingga tidak ada missing value setelah proses pembersihan. Kondisi ini ideal karena semua variabel lengkap dan tidak memerlukan imputasi, sehingga dataset siap digunakan untuk tahap modelling tanpa kendala terkait data kosong.



3.EDA

A. Melihat distribusi label death_event pada data

Diagram batang



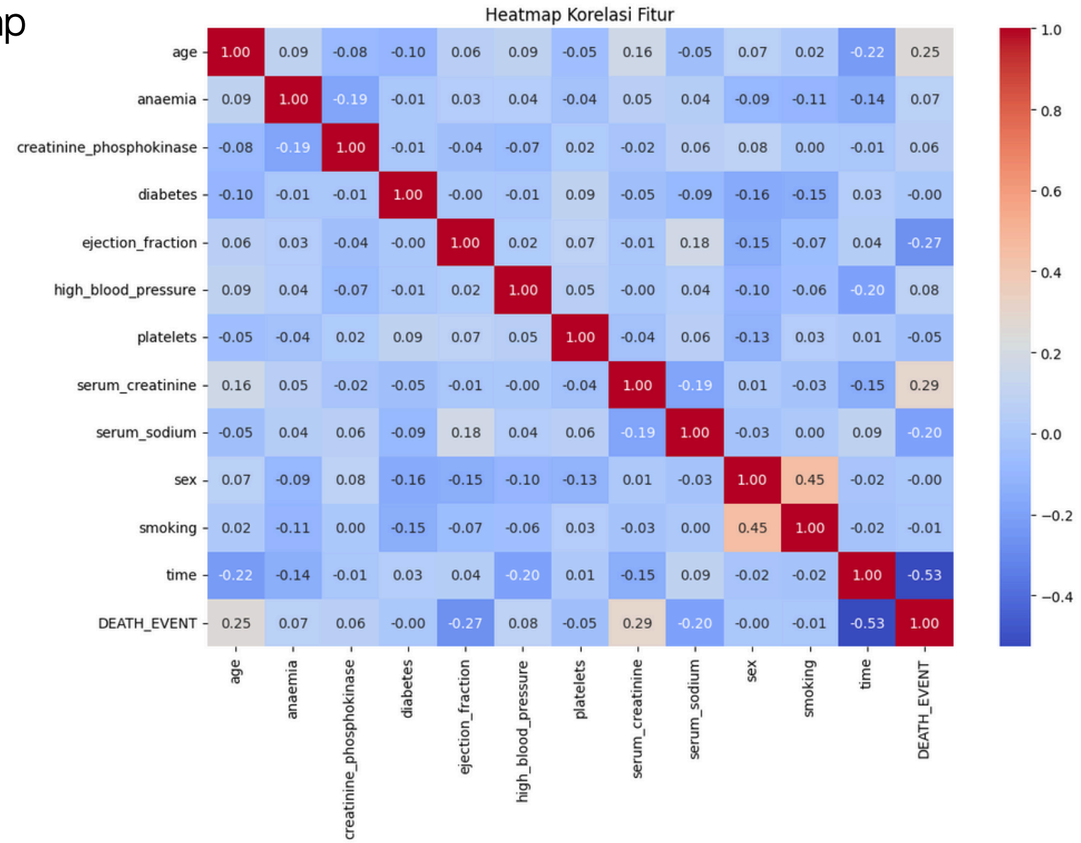
DEATH_EVENT	
0	203
1	96

Berdasarkan diagram batang serta jumlah label 0 dan 1 pada kolom death_event, diketahui bahwa sebanyak **203** pasien bertahan hidup selama periode tindak lanjut(follow up) dari total **299** pasien , atau sekitar **67,89%** pasien yang berhasil bertahan hidup.
Dari dua output tersebut, dapat dilihat pula bahwa distribusi antar labelnya masih tidak seimbang(imbalanced), sehingga ***Cross Validation*** lebih dibutuhkan dibanding *Train Test* pada umumnya.

3.EDA

B. Melihat korelasi antar variabel terhadap variabel death_event

Heatmap

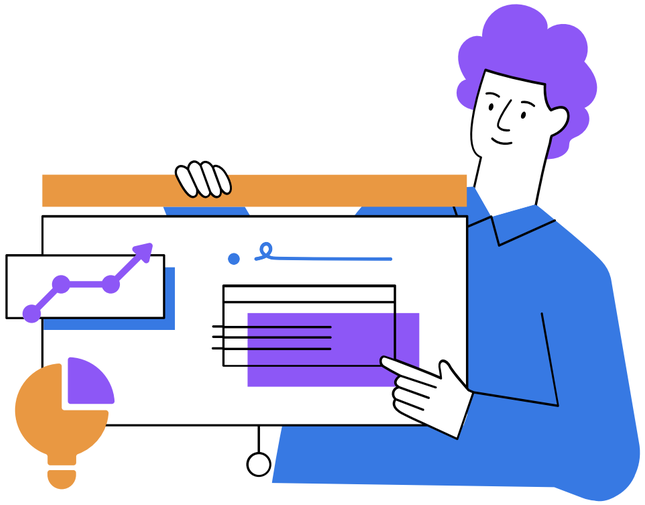


Berdasarkan heatmap di samping, terdapat beberapa informasi penting yang diketahui, yakni

1. Korelasi positif tertinggi terhadap death_event dipegang oleh variabel serum_creatine(0.29). Hal ini menandakan meningkatnya risiko kematian seiring dengan tingginya kadar kreatin serum.
2. Korelasi negatif terkecil terhadap death_event dipegang oleh variabel time(-0.53). Hal ini menandakan menurunnya risiko kematian berdasarkan lamanya waktu tindak lanjut(follow up) yang dilakukan.

Kelompok 4

4.Modelling & Evaluasi



A. Modelling - KNN Menggunakan Pipeline

Algoritma yang digunakan: K-Nearest Neighbors (KNN):
Model memprediksi *DEATH_EVENT* berdasarkan kedekatan jarak antar data.
Semakin dekat suatu data dengan kumpulan pasien tertentu, semakin besar kemungkinan dia memiliki label yang sama.

a. Pisahkan fitur dan label

```
X = df.drop("DEATH_EVENT", axis=1)
y = df["DEATH_EVENT"]
```

b. Bangun pipeline

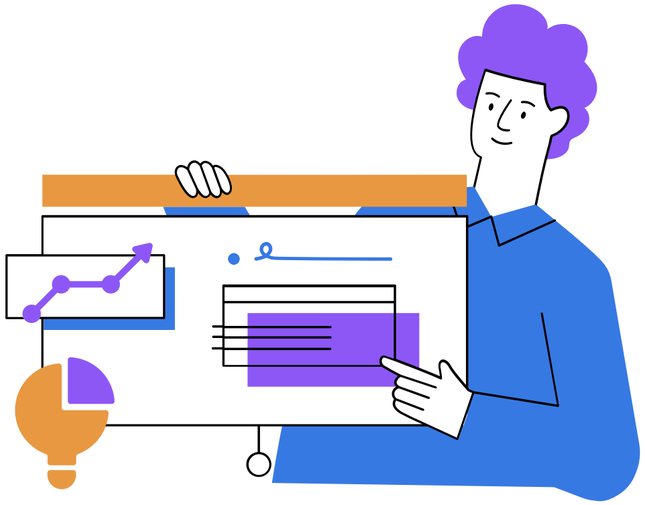
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('knn', KNeighborsClassifier())
])
```

Pipeline:

- StandardScaler: menormalkan semua fitur agar skala seragam
- KNN Classfier: algoritma jarak untuk klasifikasi

Alasan normalisasi (StandardScaler) dilakukan:
KNN menggunakan jarak (biasanya Euclidean) = fitur dengan skala besar bisa mendominasi tanpa scalling.

4.Modelling & Evaluasi



B. Pemilihan Nilai K untuk KNN

Nilai K yang diuji:

3, 4, 5, 6, 7, 8, 9, 11, 13, 15

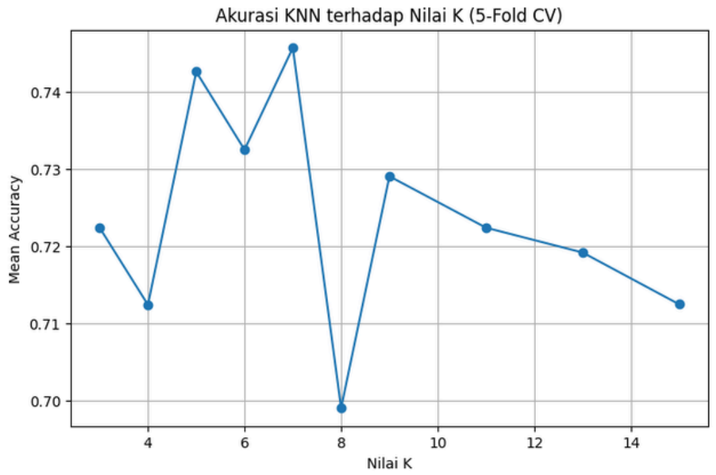
Pengaruh nilai K:

- K kecil, model sensitif, rawan overfitting
- K besar, model terlalu general, rawan underfitting

Evaluasi dilakukan melalui 5 *Fold Cross Validation*

```
for k, acc in zip(k_values, mean_scores):
    print(f"K = {k}, Mean Accuracy = {acc:.4f}")
```

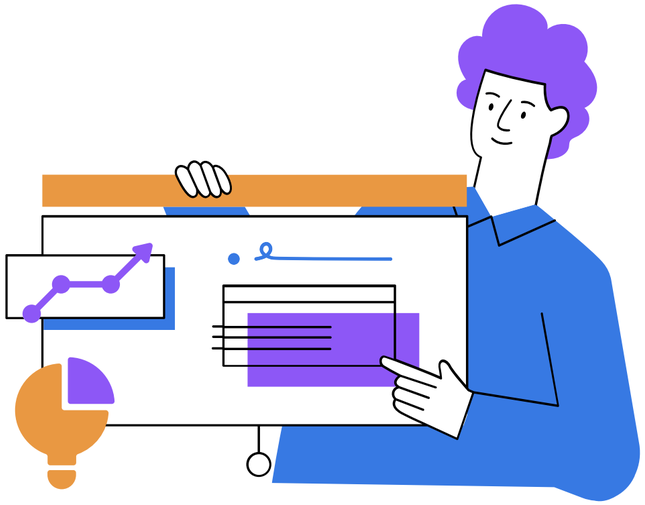
```
K = 3, Mean Accuracy = 0.7224
K = 4, Mean Accuracy = 0.7124
K = 5, Mean Accuracy = 0.7426
K = 6, Mean Accuracy = 0.7325
K = 7, Mean Accuracy = 0.7457
K = 8, Mean Accuracy = 0.6990
K = 9, Mean Accuracy = 0.7290
K = 11, Mean Accuracy = 0.7224
K = 13, Mean Accuracy = 0.7192
K = 15, Mean Accuracy = 0.7124
```



Kesimpulan:

Nilai K terbaik adalah **K=7**, dengan akurasi rata-rata tertinggi 74,57%

4.Modelling & Evaluasi



C. Output Cross Validation K terbaik KNN

Cross Validation membagi data menjadi 5 bagian (fold):
Setiap iterasi: 80% train, 20% test

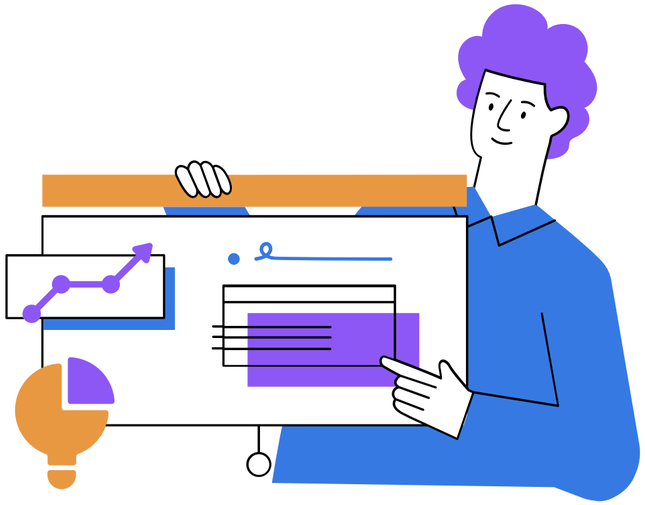
Hasil:

```
Akurasi per fold: [0.68333333 0.73333333 0.8        0.8        0.71186441]
Rata-rata: 0.7457062146892655
Standar deviasi: 0.04708367600563066
```

- **Akurasi per fold:** Lumayan stabil
- **Rata-rata:** 0.7457
- **Standar deviasi:** Cukup kecil, model bisa dibilang stabil

Cross Validation memberikan evaluasi yang lebih adil dan akurat dibandingkan train-test split tunggal.

4.Modelling & Evaluasi



D. Evaluasi Tambahan Model K Terbaik KNN

Hasil rata-rata metrik:

```
Precision per fold: [1.         0.90909091 0.625      0.71428571 0.5       ]
Mean precision: 0.7496753246753247

Recall per fold: [0.24       0.4        0.35714286 0.33333333 0.29411765]
Mean recall: 0.32491876750700277

F1-score per fold: [0.38709677 0.55555556 0.45454545 0.45454545 0.37037037]
Mean F1-score: 0.44442272184207665
```

Interpretasi:

- Precision: prediksi kematian cukup tepat
- Recall: recall rendah menunjukkan banyak kasus berisiko tidak terdeteksi
- F1 - score: menunjukkan keseimbangan keduanya masih belum optimal

Penyebab rendahnya Recall, F1-score:

Dataset inbalanced (kelas 1 jumlahnya jauh lebih sedikit dibandingkan kelas 0)

Kelompok 4

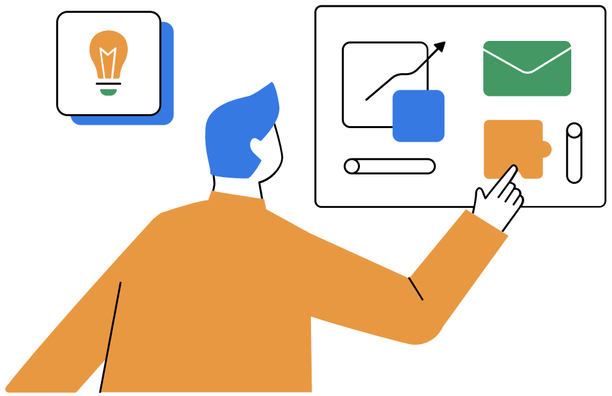
Insight & Rekomendasi

Insight

Berdasarkan analisis, pasien yang memiliki usia lebih tua, kadar serum kreatinin yang tinggi, dan waktu perawatan yang lebih cepat cenderung memiliki risiko kematian lebih besar. Ketidakseimbangan jumlah data antara pasien yang selamat dan meninggal terlihat mempengaruhi performa model, di mana KNN lebih mudah mengenali pasien yang selamat dibandingkan yang berisiko meninggal. Selain itu, model menunjukkan bahwa nilai K yang terlalu kecil membuat prediksi tidak stabil, sementara K yang terlalu besar mengaburkan perbedaan antar pasien, sehingga nilai K=7 memberikan keseimbangan terbaik.

Rekomendasi

- Bisa menggunakan teknik penyeimbangan data seperti **SMOTE** atau oversampling untuk meningkatkan nilai recall terhadap kelas DEATH_EVENT.
- Terapkan ***Stratified K-Fold*** agar proporsi kelas lebih stabil pada tiap fold.



Kesimpulan

Penelitian ini menunjukkan bahwa algoritma K-Nearest Neighbors (KNN) dapat digunakan untuk memprediksi risiko kematian pasien *Heart Failure* berdasarkan 12 fitur klinis. Setelah dilakukan preprocessing, EDA, dan pemilihan parameter menggunakan 5-*Fold Cross Validation*, model terbaik diperoleh pada nilai K = 7. Model ini menghasilkan akurasi dan *precision* yang cukup baik, namun memiliki *recall* yang rendah sehingga masih kurang optimal dalam mendeteksi pasien yang benar-benar berisiko meninggal.

Poin Kesimpulan Utama:

- KNN berhasil membangun model prediksi dengan akurasi terbaik sekitar 74.6% pada K=7.
- *Precision* tinggi (0.75) menunjukkan model cukup akurat saat memprediksi kematian.
- *Recall* rendah (0.32) menandakan model masih sering melewatkan pasien yang berisiko meninggal.
- Fitur yang paling berkaitan dengan DEATH_EVENT adalah *age*, *serum_creatinine*, dan *time*.
- Dataset terbukti imbalanced, menyebabkan evaluasi pada kelas “meninggal” menjadi kurang baik.

Model ini cukup baik sebagai baseline atau analisis awal, tetapi belum layak digunakan untuk keputusan klinis karena kemampuan pendeteksian risiko masih terbatas akibat rendahnya *recall*.

Kelompok 4

Thank You

+123-456-7890

hello@reallygreatsite.com

www.reallygreatsite.com