LAPORAN TUGAS BESAR

PERBANDINGAN ALGORITMA DECISION TREE

DAN KNN UNTUK PREDIKSI DIAGNOSIS ARRHYTHMIA

Laporan tugas besar ini disusun guna memenuhi tugas Mata Kuliah Pengantar Kecerdasan Buatan (CPI2M3) CLO-3 sebagai pengganti UAS.

PROGRAM STUDI S1 INFORMATIKA PJJ

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2023

PERNYATAAN

Disusun Oleh : Kelas : IF – 45- GAB.1 PJJ Kelompok : -	**Kontribusi Kelompok**
FARHAN RANGKUTI **1304202025**	- Membuat Program KNN - Membantu membuat laporan
MUHAMAD MEIDY MAHARDIKA **1304202024**	- Membuat Program DT - Membantu membuat laporan
KARDINA FERLINDA **1304201016**	- Membuat Laporan - Membantu Membuat Program

Tim Kami mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi. Jika melakukan plagiarism atau jenis pelanggaran lainnya, maka Tim kami bersedia diberi nilai E untuk Mata Kuliah ini.

DAFTAR ISI

BAB I

PENDAHULUAN

Arrhythmia (Gangguan irama jantung) merupakan kelainan elektrofisologi jantung yang disebabkan oleh gangguan sistem konduksi serta gangguan pembentukan dan penghantar impuls listrik. Beberapa faktor yang memengaruhi penyakit aritmia antara lain usia, tekanan darah, tinggi badan serta berat badan.

Penyakit aritmia ini dapat dikenali dengan menggunakan rekam jantung atau elektrokardiogram (EKG). Data numerik yang dihasilkan oleh EKG memiliki banyak fitur yang tidak mudah diproses secara manual. Bantuan komputer dengan teknik machine learning tertentu dapat digunakan untuk mengenali penyakit secara otomatis.

1. Data Set

Pada project ini data terdiri dari 279 atribut. Tipe data dari masing- masing atribut ini berbeda-beda. Data atribut dapat dilihat dalam tabel di bawah ini. Sumber data yang dikelola merupakan data dari *https://archive.ics.uci.edu/ml/datasets/Arrhythmia.*

No	Nama Atribut	Tipe Data Atribut	Deskripsi Atribut
1	Age	Float64	Menyatakan usia pasien dalam tahun.
2	Sex	Object	Atribut ini menyatakan jenis kelamin pasien.
3	Height	Float64	Atribut ini menggambarkan tinggi badan pasien dalam satuan cm.
4	Weight	Float64	Atribut ini menyatakan berat badan pasien dalam satuan kg.
5	QRS_duration	Float64	Atribut ini mengindikasikan durasi atau panjang gelombang QRS dalam sinyal elektrokardiogram (EKG).
6	PR_Interval	Float64	Atribut ini mencerminkan interval PR pada sinyal EKG, yaitu waktu antara gelombang P dan gelombang QRS.
7	QT_Interval	Float64	Atribut ini menggambarkan interval QT pada sinyal EKG, yaitu waktu

			antara gelombang QRS dan gelombang T.
8	T_Interval	Float64	Atribut ini menyatakan interval T pada sinyal EKG, yaitu waktu antara gelombang T dan gelombang P berikutnya.
9	P_Interval	Float64	Atribut ini mengindikasikan interval P pada sinyal EKG, yaitu waktu antara dua gelombang P.
10	QRS	Object	Atribut ini mungkin mencerminkan karakteristik atau kategori tertentu dari gelombang QRS pada sinyal EKG.
11	T	Object	Atribut ini mungkin mencerminkan karakteristik atau kategori tertentu dari gelombang T pada sinyal EKG.
12	P	Object	Atribut ini mungkin mencerminkan karakteristik atau kategori tertentu dari gelombang P pada sinyal EKG.
13	QRST	Object	Atribut ini mungkin mencerminkan karakteristik atau kategori tertentu dari kombinasi gelombang QRS dan T pada sinyal EKG.
14	J	Object	Atribut ini mungkin mencerminkan karakteristik atau kategori tertentu dari titik J pada sinyal EKG.
15	Heart_rate	Float64	Atribut ini menyatakan denyut atau kecepatan detak jantung pasien dalam satuan tertentu.
16	DI_R_Wave	Object	Atribut ini mungkin mencerminkan karakteristik atau kategori tertentu dari gelombang R pada saluran DI dalam sinyal EKG.
17	DI Intrinsic Deflections	Object	Atribut ini mungkin mencerminkan karakteristik atau kategori tertentu dari defleksi intrinsik pada saluran

			DI dalam sinyal EKG.
18	DI_S_Wave	Object	Atribut ini mungkin mencerminkan karakteristik atau kategori tertentu dari gelombang S pada saluran DI dalam sinyal EKG.

Tabel 1.1 Tabel dan Kumpulan Data

2. Alasan Pemilihan Metode

Pemilihan dua metode terbaik dari 3 metode yang dipakai yaitu Decision Tree (DT) dan K-Nearest Neighbors (kNN), karena :

- Decision Tree (Pohon Keputusan)
  Decision Tree adalah metode pembelajaran mesin yang menggunakan struktur pohon untuk membuat keputusan berdasarkan serangkaian aturan yang menggambarkan karakteristik data. Dalam konteks diagnosa aritmia, *Decision Tree* dapat membantu mengidentifikasi pola-pola pada atribut ECG yang berkaitan dengan kelainan irama jantung. Misalnya, Decision Tree dapat menentukan apakah usia, jenis kelamin, atau fitur-fitur lainnya dapat menjadi prediktor penting dalam diagnosis aritmia.
- K-Nearest Neighbors (kNN)
  K-Nearest Neighbors adalah metode klasifikasi yang memprediksi label suatu data berdasarkan mayoritas label dari k-neighbors terdekatnya dalam ruang fitur. Dalam kasus ini, kNN dapat digunakan untuk membandingkan pola ECG dari pasien dengan pasien-pasien sebelumnya yang telah terdiagnosis aritmia. Dengan melihat kesamaan pola ECG, kNN dapat mengklasifikasikan pasien baru apakah termasuk dalam kategori aritmia atau bukan.

3. *Exploratory Data Analysis* (EDA) dan Preprocessing Data

EDA DATA EXPLORATION

```
[ ] arrhythmia_name_excel = '/content/Atribut_name.xlsx'
    output_arrhythmia_name_latest = '/content/Atribut_name.names'

    df_name = pd.read_excel(arrhythmia_name_excel)
    df_name
```

	atribut_name	tipe_data
0	Age	int64
1	Sex	object
2	Height	float64
3	Weight	float64
4	QRS_duration	float64
5	PR_interval	float64
6	QT_interval	float64
7	T_interval	float64
8	P_interval	float64
9	QRS	object
10	T	object
11	P	object
12	QRST	object
13	J	object
14	Heart_rate	object
15	DI_Q_Wave	float64
16	DI_R_Wave	float64
17	DI_S_Wave	float64
18	DI_RP_Wave	float64
19	DI_SP_Wave	float64
20	DI_IntrinsicDeflections	float64
21	DI_RR_WaveExists	object
22	DI_DD_RR_WaveExists	object
23	DI_RP_WaveExists	object

Gambar 1.1

Melakukan read file excel yang berisi nama header dan tipe data setiap kolom data frame.

```
[ ] names_content = ''
    for index, row in df_name.iterrows():
        names_content += f'@attribute {row["atribut_name"]} {row["tipe_data"]}\n'

    with open(output_arrhythmia_name_latest, 'w') as file:
        file.write(names_content)
```

```
[ ] arrhythmia = "arrhythmia.data"
    arrhythmia_name_fix = "Atribut_name.names"

    df_set = pd.read_csv(arrhythmia, header=None)

    with open(arrhythmia_name_fix, "r") as file:
        lines = file.readlines()

    atribut_lines = [line.strip() for line in lines if line.startswith("@attribute")]
    atribut_info = [line.split() for line in atribut_lines]
```

```
[ ] atribut_names = [info[1] for info in atribut_info]
    atribut_types = [info[2] for info in atribut_info]
```

```
[ ] df_set.columns = atribut_names
    df_set = df_set.astype(dict(zip(atribut_names, atribut_types)))
```

```
[ ] print(df_set)
```

Gambar 1.2

Menggabungkan data agar mempunyai header dan memberikan nama kolom dan mengatur tipe data pada Data frame.

```
df_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 452 entries, 0 to 451
Columns: 280 entries, Age to Diagnosa
dtypes: float64(265), int64(1), object(14)
memory usage: 988.9+ KB
```

```
df_set.describe()
```

	Age	Height	Weight	QRS_duration	PR_interval	QT_interval	T_interval	P_interval	DI_Q_Wave	DI_R_Wave
count	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000
mean	46.471239	166.188053	68.170354	88.920354	155.152655	367.207965	169.949115	90.004425	5.628319	51.628319
std	16.466631	37.170340	16.590803	15.364394	44.842283	33.385421	35.633072	25.826643	10.650001	18.249901
min	0.000000	105.000000	6.000000	55.000000	0.000000	232.000000	108.000000	0.000000	0.000000	0.000000
25%	36.000000	160.000000	59.000000	80.000000	142.000000	350.000000	148.000000	79.000000	0.000000	40.000000
50%	47.000000	164.000000	68.000000	86.000000	157.000000	367.000000	162.000000	91.000000	0.000000	48.000000
75%	58.000000	170.000000	79.000000	94.000000	175.000000	384.000000	179.000000	102.000000	12.000000	60.000000
max	83.000000	780.000000	176.000000	188.000000	524.000000	509.000000	381.000000	205.000000	88.000000	156.000000

```
total_rows, total_attributes = df_set.shape
print('Jumlah data:', total_rows)
print("Jumlah atribut:", total_attributes)
df_set.head()
```

```
Jumlah data: 452
Jumlah atribut: 280
```

	Age	Sex	Height	Weight	QRS_duration	PR_interval	QT_interval	T_interval	P_interval	QRS	T	P	QRST	J	Heart_rate	D
0	75	0	190.0	80.0	91.0	193.0	371.0	174.0	121.0	-16	13	64	-2	?	63	
1	56	1	165.0	64.0	81.0	174.0	401.0	149.0	39.0	25	37	-17	31	?	53	
2	54	0	172.0	95.0	138.0	163.0	386.0	185.0	102.0	96	34	70	66	23	75	
3	55	0	175.0	94.0	100.0	202.0	380.0	179.0	143.0	28	11	-5	20	?	71	
4	75	0	190.0	80.0	88.0	181.0	360.0	177.0	103.0	-16	13	61	3	?	?	

Gambar 1.3

Memanggil .info(), .describe(), .head(), .tail() untuk melihat info-info yang ada pada data.

```
df_set.tail()
```

	Age	Sex	Height	Weight	QRS_duration	PR_interval	QT_interval	T_interval	P_interval	QRS	T	P	QRST	J	Heart_rate	DI_Q_Wave	DI_R_Wave	DI_S_Wave	DI_RP_Wave	DI_SP_Wave
447	53	1	160.0	70.0	80.0	199.0	382.0	154.0	117.0	-37	4	40	-27	?	63	0.0	52.0	24.0	0.0	0.0
448	37	0	190.0	85.0	100.0	137.0	361.0	201.0	73.0	86	66	52	79	?	73	0.0	44.0	36.0	0.0	0.0
449	36	0	166.0	68.0	108.0	176.0	365.0	194.0	116.0	-85	-19	-61	-70	84	84	16.0	40.0	40.0	0.0	0.0
450	32	1	155.0	55.0	93.0	106.0	386.0	218.0	63.0	54	29	-22	43	103	80	0.0	56.0	0.0	0.0	0.0
451	78	1	160.0	70.0	79.0	127.0	364.0	138.0	78.0	28	79	52	47	?	75	0.0	44.0	28.0	0.0	0.0

```
unique_class = sorted([i for i in df_set['Diagnosa'].unique()])
unique_class
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 15, 16]
```

```
Data_Object = [i for i in df_set.columns if df_set[i].dtypes == 'O']
Data_Linear = [i for i in df_set.columns if df_set[i].dtypes != "O"]
print(Data_Object)
print(Data_Linear)
```

```
['Sex', 'QRS', 'T', 'P', 'QRST', 'J', 'Heart_rate', 'DI_RR_WaveExists', 'DI_DD_RR_WaveExists', 'DI_RP_WaveExists', 'DI_DD_RP_WaveExists', 'DI_RT_WaveExists', 'DI_DD_RT_WaveExists', 'Diag
['Age', 'Height', 'Weight', 'QRS_duration', 'PR_interval', 'QT_interval', 'T_interval', 'P_interval', 'DI_Q_Wave', 'DI_R_Wave', 'DI_S_Wave', 'DI_RP_Wave', 'DI_SP_Wave', 'DI_IntrinsicDefl
```

```
df_set = df_set.replace(to_replace="?", value=None)
```

```
df_arrhythmia = df_set[('J')]
df_arrhythmia.head()
```

```
0    None
1    None
2      23
3    None
4    None
Name: J, dtype: object
```

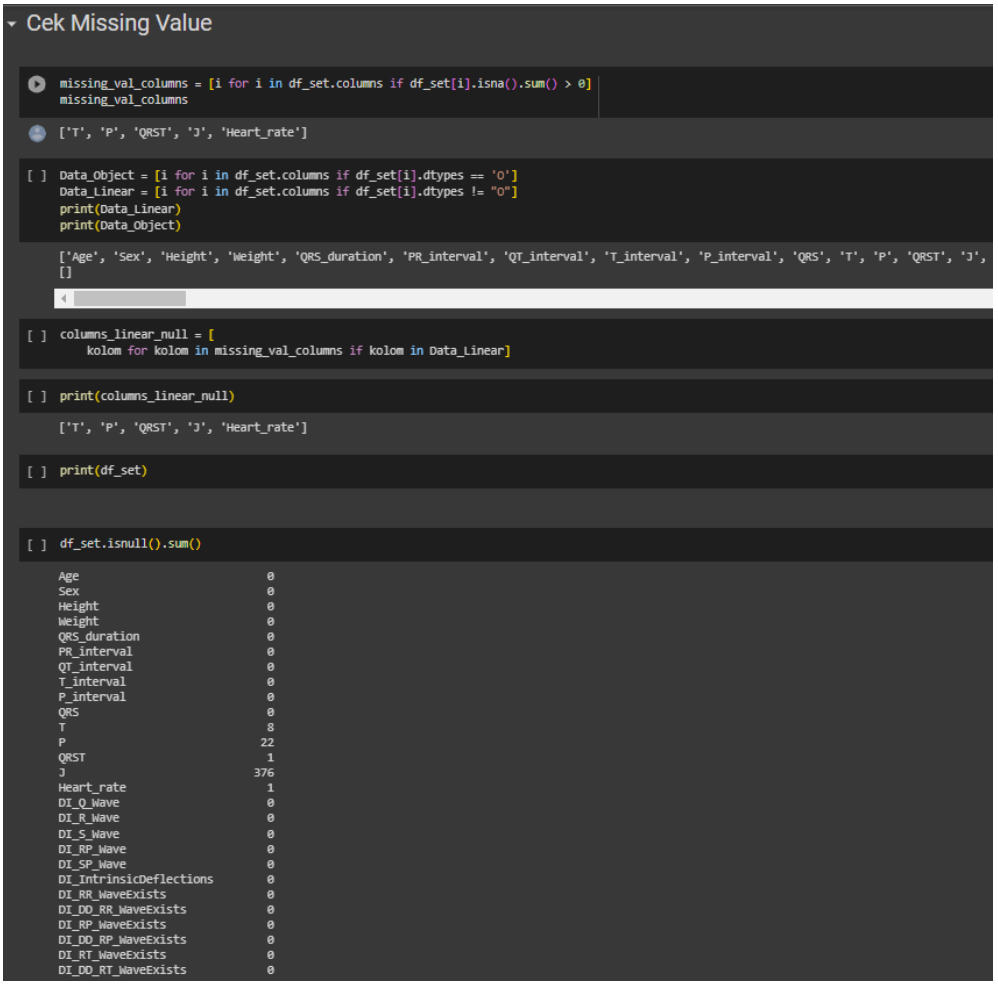
```
df_set[Data_Object] = df_set[Data_Object].astype(float)
```

```
df_set[Data_Object].dtypes
```

```
Sex                 float64
QRS                 float64
T                   float64
P                   float64
QRST                float64
J                   float64
Heart_rate          float64
DI_RR_WaveExists    float64
DI_DD_RR_WaveExists float64
DI_RP_WaveExists    float64
DI_DD_RP_WaveExists float64
DI_RT_WaveExists    float64
```

Gambar 1.3

Melakukan output pada variabel unique_class yang berisi data pada kolom diagnosa yang sudah disort. Selanjutnya melakukan pemisahan antara kolom yang berisi data Linear (float atau integer) dan data yang berisi object. Selanjutnya melakukan replace pada data yang hilang (?) dengan None (null). Setelah itu mengubah kolom pada df_set yang berisi tipe data object menjadi float kemudian melakukan output agar hasilnya bisa dilihat.



Gambar 1. 4

Melakukan pengecekan pada kolom data frame manakala ada nilai Null atau NaN dan menyimpan list kolom yang mempunyai nilai Null tersebut ke dalam variabel missing_val_columns. Selanjutnya melakukan pengecekan ulang pada variabel Data_object untuk mengecek apakah tipe data sebelumnya yaitu object sudah benar diubah ke tipe data float. Kemudian melakukan pengecekan pada Data_Linear apakah kolom yang ada di missing_val_columns ada pada Data_Linear dan disimpan pada variabel baru yaitu columns_linear_null. Kemudian kita akan melakukan penjumlahan nilai null pada setiap kolom dan mengeluarkan outputnya.

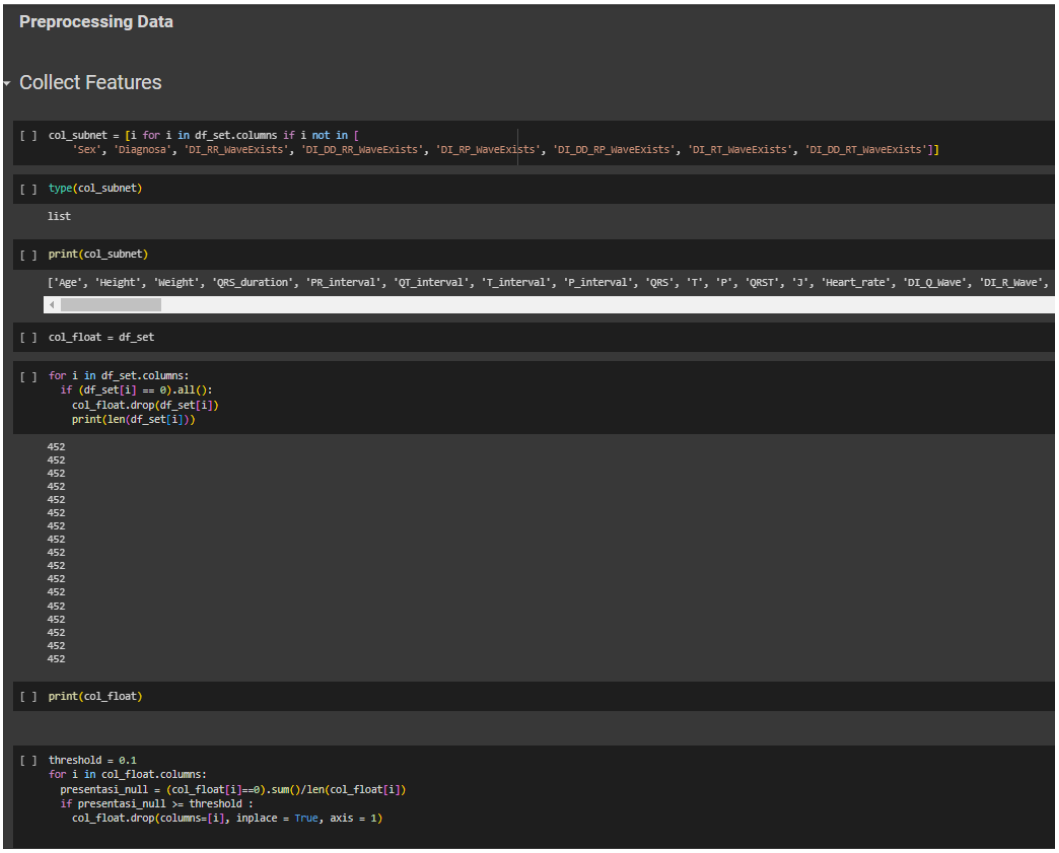
```
[ ]  df_set = df_set.fillna(df_set.mean())
     print(df_set)
```

Gambar 1.5

	P_interval	QRS	T	P	QRST	J
0	121.0	-16.0	13.000000	64.000000	-2.000000	-13.592105
1	39.0	25.0	37.000000	-17.000000	31.000000	-13.592105
2	102.0	96.0	34.000000	70.000000	66.000000	23.000000
3	143.0	28.0	11.000000	-5.000000	20.000000	-13.592105
4	103.0	-16.0	13.000000	61.000000	3.000000	-13.592105
5	91.0	107.0	66.000000	52.000000	88.000000	-13.592105
6	77.0	77.0	49.000000	75.000000	65.000000	-13.592105
7	70.0	67.0	7.000000	8.000000	51.000000	-13.592105
8	63.0	61.0	69.000000	78.000000	66.000000	84.000000
9	73.0	85.0	34.000000	70.000000	71.000000	-13.592105
10	83.0	72.0	71.000000	68.000000	72.000000	-13.592105
11	65.0	12.0	37.000000	49.000000	26.000000	-13.592105
12	81.0	-24.0	42.000000	41.000000	-13.000000	-13.592105
13	104.0	68.0	51.000000	60.000000	63.000000	-13.592105
14	94.0	46.0	20.000000	45.000000	40.000000	-13.592105
15	65.0	36.0	45.000000	68.000000	40.000000	-13.592105
16	61.0	77.0	75.000000	77.000000	75.000000	-13.592105
17	52.0	57.0	49.000000	-2.000000	54.000000	-13.592105
18	83.0	73.0	-24.000000	61.000000	42.000000	-13.592105
19	79.0	-12.0	28.000000	50.000000	1.000000	-13.592105
20	183.0	50.0	39.000000	46.000000	43.000000	-13.592105
21	78.0	81.0	78.000000	67.000000	80.000000	-13.592105
22	82.0	62.0	56.000000	45.000000	60.000000	-13.592105
23	92.0	4.0	10.000000	58.000000	5.000000	-13.592105
24	82.0	52.0	17.000000	105.000000	42.000000	-13.592105
25	60.0	-34.0	112.000000	154.000000	7.000000	-13.592105
26	125.0	90.0	52.000000	60.000000	77.000000	-13.592105
27	83.0	10.0	48.000000	39.000000	30.000000	-13.592105
28	99.0	-8.0	153.000000	41.000000	0.000000	-13.592105
29	82.0	-37.0	172.000000	-5.000000	-67.000000	160.000000
30	91.0	-52.0	16.000000	54.000000	-32.000000	-13.592105

Gambar 1.6

Melakukan pengisian pada kolom yang memiliki nilai Null dan diisi dengan nilai rata-rata (mean) dari kolom tersebut.



Gambar 1.7

Selanjutnya melakukan looping pada data frame agar menyimpan nama kolom kecuali kolom yang bertipe object dan disimpan pada variabel col_subnet. Kemudian melakukan deklarasi variabel baru yang berisi data yang sama dengan df_set untuk digunakan pada loop agar menghapus kolom yang berisi nilai 0 untuk semua barisnya.

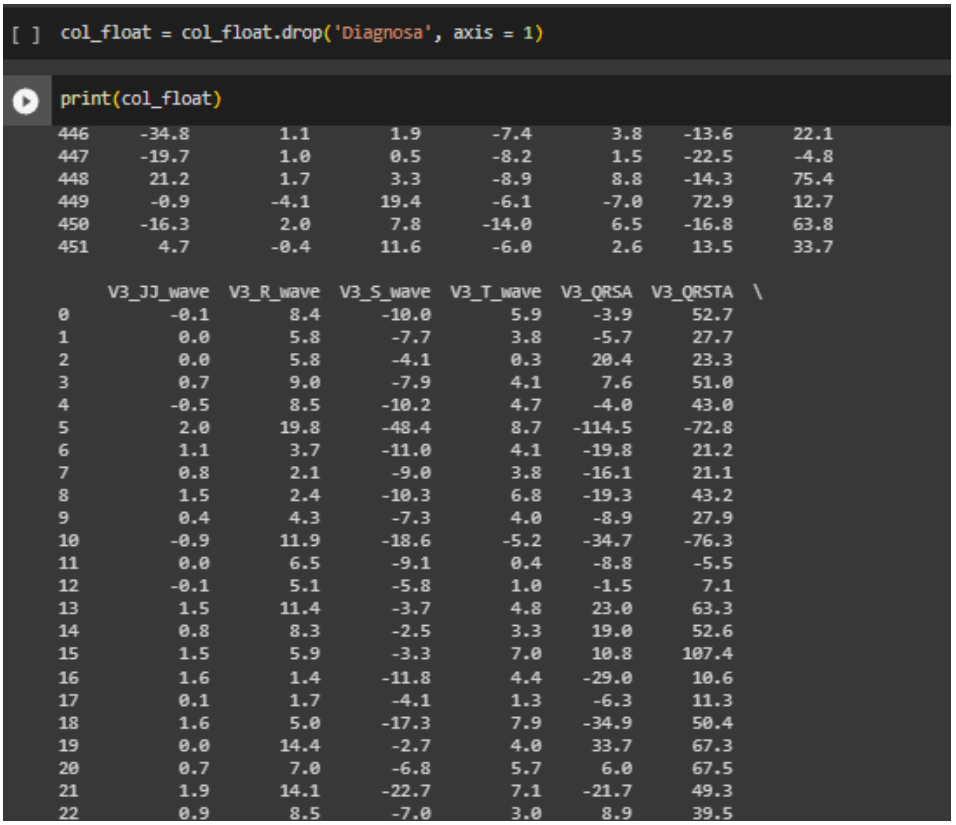
```
threshold = 0.1
for i in col_float.columns:
  presentasi_null = (col_float[i]==0).sum()/len(col_float[i])
  if presentasi_null >= threshold :
    col_float.drop(columns=[i], inplace = True, axis = 1)
```

```
[ ]  print(col_float)
```

```
    Age  Height  Weight  QRS_duration  PR_interval  QT_interval  T_interval  \
0    75   190.0    80.0          91.0        193.0        371.0       174.0
1    56   165.0    64.0          81.0        174.0        401.0       149.0
2    54   172.0    95.0         138.0        163.0        386.0       185.0
3    55   175.0    94.0         100.0        202.0        380.0       179.0
4    75   190.0    80.0          88.0        181.0        360.0       177.0
5    13   169.0    51.0         100.0        167.0        321.0       174.0
6    40   160.0    52.0          77.0        129.0        377.0       133.0
7    49   162.0    54.0          78.0          0.0        376.0       157.0
8    44   168.0    56.0          84.0        118.0        354.0       160.0
9    50   167.0    67.0          89.0        130.0        383.0       156.0
10   62   170.0    72.0         102.0        135.0        401.0       156.0
11   45   165.0    86.0          77.0        143.0        373.0       150.0
12   54   172.0    58.0          78.0        155.0        382.0       163.0
13   30   170.0    73.0          91.0        180.0        355.0       157.0
14   44   160.0    88.0          77.0        158.0        399.0       163.0
15   47   150.0    48.0          75.0        132.0        350.0       169.0
16   47   171.0    59.0          82.0        145.0        347.0       169.0
17   46   158.0    58.0          70.0        120.0        353.0       122.0
18   73   165.0    63.0          91.0        154.0        392.0       175.0
19   57   166.0    72.0          82.0        181.0        399.0       158.0
20   28   160.0    58.0          83.0        251.0        383.0       189.0
21   45   169.0    67.0          90.0        122.0        336.0       177.0
22   36   153.0    75.0          71.0        132.0        364.0       169.0
23   57   165.0    59.0          75.0        157.0        406.0       143.0
24   40   153.0    55.0          82.0        140.0        388.0       149.0
25   44   169.0    80.0         109.0        128.0        382.0       195.0
26   34   170.0    73.0          94.0        186.0        373.0       224.0
27   31   160.0    54.0          95.0        161.0        407.0       168.0
28   56   164.0    65.0          90.0        164.0        420.0       381.0
29   51   160.0    83.0          96.0        147.0        400.0       301.0
30   53   175.0    85.0          85.0        157.0        408.0       172.0
31   58   163.0    68.0          71.0        136.0        339.0       152.0
32   50   160.0    73.0          75.0        125.0        353.0       183.0
33   52   155.0    70.0          78.0        137.0        368.0       148.0
34   69   176.0    75.0          82.0        145.0        357.0       129.0
35   44   160.0    45.0          69.0        178.0        357.0       137.0
36   50   172.0    80.0         103.0        142.0        366.0       161.0
37   35   164.0    94.0          85.0        200.0        385.0       174.0
38   62   163.0    60.0          80.0        185.0        354.0       166.0
39   45   175.0    80.0          94.0        163.0        401.0       159.0
40   43   157.0    71.0          80.0        162.0        383.0       141.0
41   40   160.0    75.0          79.0        154.0        350.0       160.0
42   30   160.0    62.0          92.0        154.0        346.0       158.0
43   34   165.0    61.0          84.0        152.0        383.0       144.0
44   40   160.0    51.0          86.0        177.0        367.0       147.0
45   75   156.0    55.0          73.0        159.0        350.0       138.0
46   69   160.0    71.0          75.0        156.0        322.0       172.0
47   30   158.0    57.0          73.0        137.0        369.0       143.0
48   41   159.0    55.0          78.0        228.0        429.0       130.0
49   34   159.0    68.0          80.0        135.0        379.0       149.0
```

Gambar 1.8

Melakukan penghapusan kolom yang memiliki persentase nilai 0 di dalamnya lebih dari **10%** , Kemudian memanggil output col_float yang memperlihatkan data sudah berubah.

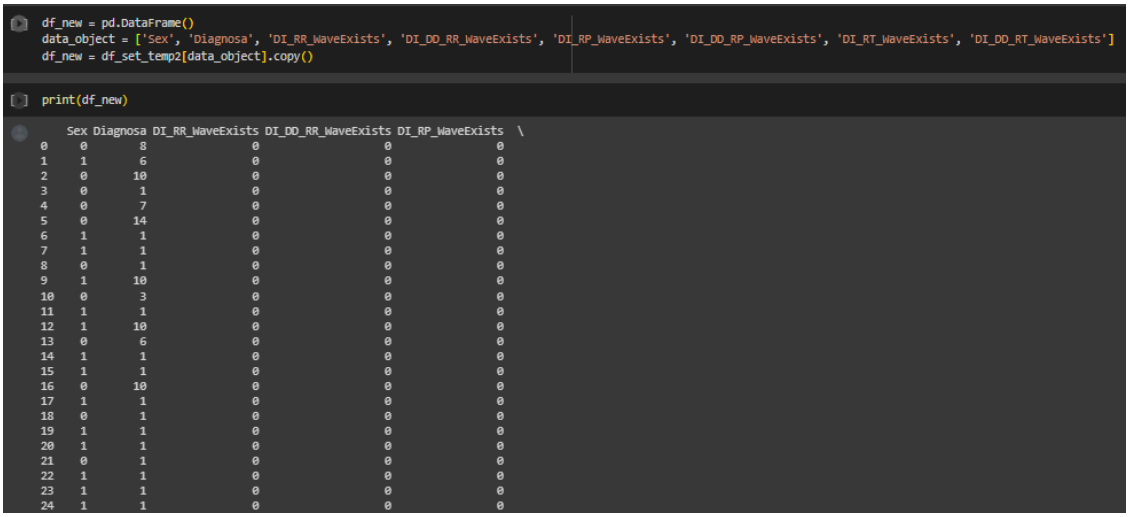


```
[ ]  col_float = col_float.drop('Diagnosa', axis = 1)

     print(col_float)
```

```
     -34.8   1.1   1.9   -7.4   3.8   -13.6   22.1
     -19.7   1.0   0.5   -8.2   1.5   -22.5   -4.8
      21.2   1.7   3.3   -8.9   8.8   -14.3   75.4
      -0.9  -4.1  19.4   -6.1  -7.0    72.9   12.7
     -16.3   2.0   7.8  -14.0   6.5   -16.8   63.8
       4.7  -0.4  11.6   -6.0   2.6    13.5   33.7

     V3_JJ_wave  V3_R_wave  V3_S_wave  V3_T_wave  V3_QRSA  V3_QRSTA  \
0          -0.1        8.4      -10.0        5.9     -3.9      52.7
1           0.0        5.8       -7.7        3.8     -5.7      27.7
2           0.0        5.8       -4.1        0.3     20.4      23.3
3           0.7        9.0       -7.9        4.1      7.6      51.0
4          -0.5        8.5      -10.2        4.7     -4.0      43.0
5           2.0       19.8      -48.4        8.7   -114.5     -72.8
6           1.1        3.7      -11.0        4.1     -19.8      21.2
7           0.8        2.1       -9.0        3.8     -16.1      21.1
8           1.5        2.4      -10.3        6.8     -19.3      43.2
9           0.4        4.3       -7.3        4.0      -8.9      27.9
10         -0.9       11.9      -18.6       -5.2     -34.7     -76.3
11          0.0        6.5       -9.1        0.4      -8.8      -5.5
12         -0.1        5.1       -5.8        1.0      -1.5       7.1
13          1.5       11.4       -3.7        4.8      23.0      63.3
14          0.8        8.3       -2.5        3.3      19.0      52.6
15          1.5        5.9       -3.3        7.0      10.8     107.4
16          1.6        1.4      -11.8        4.4     -29.0      10.6
17          0.1        1.7       -4.1        1.3      -6.3      11.3
18          1.6        5.0      -17.3        7.9     -34.9      50.4
19          0.0       14.4       -2.7        4.0      33.7      67.3
20          0.7        7.0       -6.8        5.7       6.0      67.5
21          1.9       14.1      -22.7        7.1     -21.7      49.3
22          0.9        8.5       -7.0        3.0       8.9      39.5
```

Gambar 1.9

Melakukan drop pada kolom Diagnosa kemudian mengeluarkan outputnya.



```
df_new = pd.DataFrame()
data_object = ['Sex', 'Diagnosa', 'DI_RR_WaveExists', 'DI_DD_RR_WaveExists', 'DI_RP_WaveExists', 'DI_DD_RP_WaveExists', 'DI_RT_WaveExists', 'DI_DD_RT_WaveExists']
df_new = df_set_temp2[data_object].copy()

print(df_new)
```

```
    Sex Diagnosa DI_RR_WaveExists DI_DD_RR_WaveExists DI_RP_WaveExists  \
0     0        8                0                   0                0
1     1        6                0                   0                0
2     0       10                0                   0                0
3     0        1                0                   0                0
4     0        7                0                   0                0
5     0       14                0                   0                0
6     1        1                0                   0                0
7     1        1                0                   0                0
8     0        1                0                   0                0
9     1       10                0                   0                0
10    0        3                0                   0                0
11    1        1                0                   0                0
12    1       10                0                   0                0
13    0        6                0                   0                0
14    1        1                0                   0                0
15    1        1                0                   0                0
16    0       10                0                   0                0
17    1        1                0                   0                0
18    0        1                0                   0                0
19    1        1                0                   0                0
20    1        1                0                   0                0
21    0        1                0                   0                0
22    1        1                0                   0                0
23    1        1                0                   0                0
24    1        1                0                   0                0
```

Gambar 1.10

Membuat data frame kosong pada variabel df_new dan mengisi variabel tersebut dengan 8 kolom bertipe object yang sudah dipaparkan sebelumnya.

```
[ ] df_new = df_new.drop(['DI_RR_WaveExists', 'DI_DD_RR_WaveExists', 'DI_RP_WaveExists', 'DI_DD_RP_WaveExists', 'DI_RT_WaveExists', 'DI_DD_RT_WaveExists'], axis=1)

[ ] print(df_new)

    Sex Diagnosa
0     0        8
1     1        6
2     0       10
3     0        1
4     0        7
5     0       14
6     1        1
7     1        1
8     0        1
9     1       10
10    0        3
11    1        1
12    1       10
13    0        6
14    1        1
15    1        1
16    0       10
17    1        1
18    0        1
19    1        1
20    1        1
21    0        1
22    1        1
23    1        1
24    1        1
25    0       16
26    0       14
27    1       10
28    1        2
29    1        2
30    0        6
31    1        1
32    1        1
```

Gambar 1.11

Melakukan drop pada kolom yang berisi data kosong sehingga hanya tersisa kolom ‘Sex’ dan ‘Diagnosa’

```
[ ] df_fix = pd.concat([col_float, df_new], axis = 1)

print(df_fix)

    V6_R_wave  V6_P_wave  V6_T_wave  V6_QRSA  V6_QRSTA  Sex  Diagnosa
0         9.0        0.9        2.9     23.3      49.4    0         8
1         8.5        0.2        2.1     20.4      38.8    1         6
2         9.5        0.3        3.4     12.3      49.0    0        10
3        12.2        0.4        2.6     34.6      61.6    0         1
4        13.1       -0.1        3.9     25.4      62.8    0         7
5        12.2        0.9        2.2     13.5      31.1    0        14
6         6.5        0.4        1.0     14.3      20.5    1         1
7         8.2        0.1        0.5     15.8      19.8    1         1
8         7.0        0.6        2.1     12.5      30.9    0         1
9        10.8        0.8        0.9     20.1      25.1    1        10
10        9.0        0.8        0.9     12.3      19.3    0         3
11        4.4        0.5        1.5      4.9      17.2    1         1
12        6.3        0.8        0.5      8.8      12.1    1        10
13       12.3        0.4        2.1     28.5      48.6    0         6
14       12.4        0.3        1.7     39.2      54.1    1         1
15        7.7        0.6        1.7     17.2      31.1    1         1
16        9.4        0.6        2.3     19.5      41.1    0        10
17        6.6        0.3        0.7     17.1      20.8    1         1
18        5.7        0.4        0.5     18.2      22.4    0         1
19        7.7        0.5        1.8     25.2      38.5    1         1
20        9.1        0.6        3.3     17.1      54.7    1         1
21        8.3        0.8        1.1     11.7      19.6    0         1
22        8.9        0.5        1.7     19.7      34.3    1         1
23        6.7        0.4        1.1     18.4      28.9    1         1
24       13.6        0.5        2.5     35.3      57.3    1         1
25        6.9        0.4        1.3     20.7      29.2    0        16
26       15.3        0.6        2.6     44.0      68.4    0        14
27       12.7        0.3        3.2     25.4      54.8    1        10
28        5.4        0.4       -1.4     17.2       3.0    1         2
```

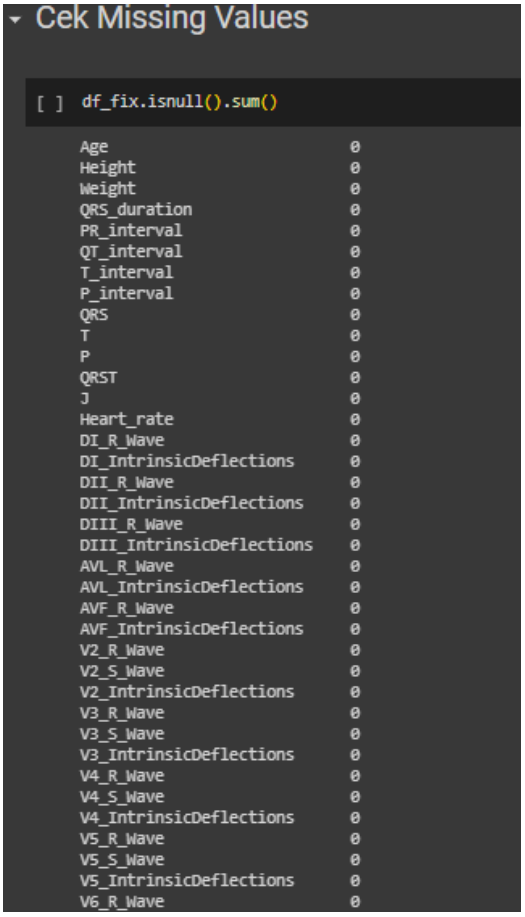
Gambar 1.12

Melakukan concat pada data frame col_float dan df_new sehingga didapatkan data frame yang sudah bersih.

```
[ ] df_fix.duplicated(keep=False).sum()

    0
```

Gambar 1.13

Melakukan pengecekan apabila ada kolom yang terduplikat.



Gambar 1.14

Melakukan pengecekan apabila masih ada missing values pada kolom data frame df_fix.

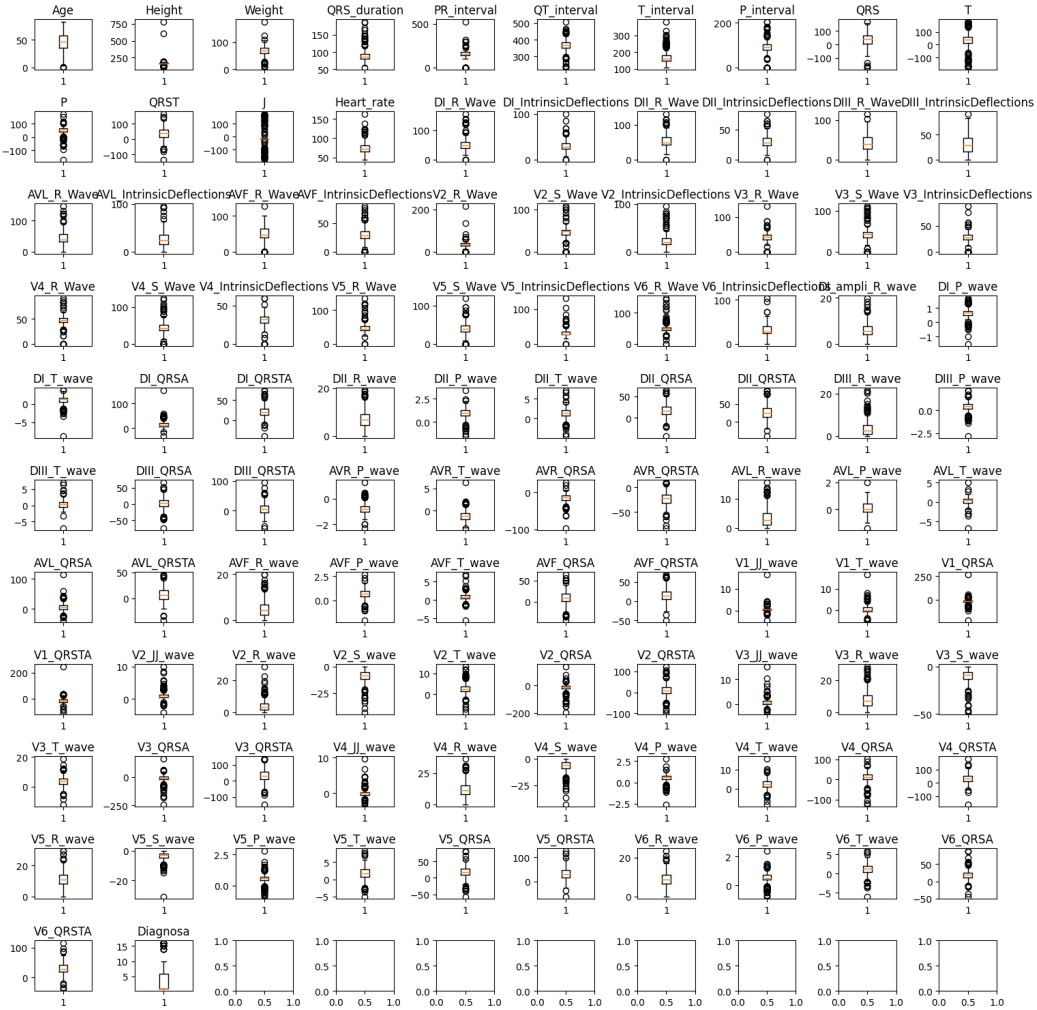
Gambar 1.15

Melakukan deklarasi variabel allColumn yang berisi semua nama kolom kecuali Index, Sex, dan Class yang berisi sebanyak 102 kolom.

```
fig, ax = plt.subplots(nrows=11, ncols=10, figsize=(15, 15))
row = 0
col = 0
for i, kolom in enumerate(allColumn[0:102]):
    if col == 10 and row < 11:
        col = 0
        row += 1

    ax[row][col].boxplot(df_fix[kolom])
    ax[row][col].set_title(kolom)

    col += 1

plt.tight_layout()
plt.show()
```

Gambar 1.16

Melakukan plotting outlier pada kolom yang ada di allColumn.



Gambar 1.17

Hasil plotting outlier kolom yang ada di variabel allColumn.

```
correlation = df_fix.corr().abs().iloc[:-1, -1]

print(correlation)
```

```
Age                          0.199728
Height                       0.092235
Weight                       0.050682
QRS_duration                 0.082791
PR_interval                  0.021048
QT_interval                  0.150979
T_interval                   0.014430
P_interval                   0.004283
QRS                          0.255203
T                            0.078167
P                            0.012089
QRST                         0.211305
J                            0.071079
Heart_rate                   0.233349
DI_R_Wave                    0.159676
DI_IntrinsicDeflections      0.072924
DII_R_Wave                   0.255147
DII_IntrinsicDeflections     0.099488
DIII_R_Wave                  0.251178
DIII_IntrinsicDeflections    0.118152
AVL_R_Wave                   0.103802
AVL_IntrinsicDeflections     0.108342
AVF_R_Wave                   0.302394
AVF_IntrinsicDeflections     0.131049
V2_R_Wave                    0.075253
V2_S_Wave                    0.129169
V2_IntrinsicDeflections      0.028533
V3_R_Wave                    0.233594
V3_S_Wave                    0.114379
V3_IntrinsicDeflections      0.155942
V4_R_Wave                    0.342360
V4_S_Wave                    0.307192
V4_IntrinsicDeflections      0.265427
```

Gambar 1.18

Melakukan korelasi pada setiap data dan mengeluarkan outputnya

```
df_fix = df_fix.sample(452).reset_index(drop=True)
df_fix
```

	Age	Height	Weight	QRS_duration	PR_interval	QT_interval	T_interval	P_interval	QRS	T	P	QRST	J	Heart_rate	DI_R_Wave	DI_IntrinsicDeflections	DII_R_Wave	DII_IntrinsicDeflections	D
0	18	180.0	63.0	93.0	160.0	321.0	172.0	101.0	111.0	72.000000	75.000000	80.000000	-13.592105	92.000000	32.0	20.0	32.0	36.0	
1	57	175.0	70.0	94.0	148.0	382.0	147.0	100.0	46.0	-8.000000	76.000000	24.000000	-13.592105	72.000000	60.0	20.0	40.0	40.0	
2	17	170.0	61.0	97.0	0.0	353.0	141.0	0.0	90.0	-92.000000	48.913953	92.000000	-95.000000	86.000000	44.0	28.0	40.0	56.0	
3	35	165.0	59.0	93.0	184.0	367.0	173.0	90.0	82.0	52.000000	53.000000	69.000000	-13.592105	54.000000	44.0	24.0	56.0	32.0	
4	54	172.0	95.0	138.0	163.0	386.0	185.0	102.0	96.0	34.000000	70.000000	66.000000	23.000000	75.000000	40.0	24.0	56.0	40.0	
5	27	167.0	48.0	80.0	152.0	343.0	164.0	96.0	77.0	-68.000000	74.000000	47.000000	-13.592105	89.000000	52.0	32.0	44.0	44.0	
6	63	174.0	79.0	91.0	151.0	410.0	198.0	86.0	16.0	36.150901	2.000000	16.000000	-13.592105	59.000000	48.0	24.0	48.0	32.0	
7	18	175.0	60.0	102.0	135.0	379.0	167.0	73.0	91.0	38.000000	61.000000	69.000000	-13.592105	62.000000	40.0	40.0	44.0	40.0	
8	46	162.0	65.0	68.0	175.0	391.0	147.0	114.0	81.0	57.000000	65.000000	74.000000	-13.592105	60.000000	40.0	20.0	72.0	28.0	
9	47	162.0	74.0	102.0	126.0	357.0	135.0	83.0	55.0	81.000000	47.000000	76.000000	-13.592105	68.000000	44.0	28.0	48.0	24.0	
10	64	160.0	60.0	87.0	131.0	396.0	169.0	76.0	-4.0	-4.000000	18.000000	-4.000000	-13.592105	61.000000	56.0	36.0	48.0	28.0	
11	63	163.0	62.0	79.0	160.0	371.0	171.0	100.0	9.0	10.000000	56.000000	9.000000	-13.592105	75.000000	44.0	24.0	40.0	24.0	
12	42	178.0	75.0	87.0	136.0	356.0	202.0	87.0	-25.0	11.000000	60.000000	-6.000000	-13.592105	75.000000	44.0	20.0	48.0	32.0	

Gambar 1.19

Melakukan shuffle pada data.

Folding Data

+ Code + Text

```
fix1 =(df_fix.iloc[0:150].reset_index(drop=True), df_fix.iloc[150:1150].reset_index(drop=True))
fix2 =(df_fix.iloc[150:300].reset_index(drop=True), pd.concat([df_fix.iloc[0:150], df_fix.iloc[300:]]).reset_index(drop=True))
fix3 =(df_fix.iloc[300:].reset_index(drop=True), df_fix.iloc[0:300].reset_index(drop=True))

fix_test, fix_train = fix2
print(fix_train)
```

```
     Age  Height  Weight  QRS_duration  PR_interval  QT_interval  T_interval  \
0     18   180.0    63.0          93.0        160.0        321.0       172.0
1     57   175.0    70.0          94.0        148.0        382.0       147.0
2     17   170.0    61.0          97.0          0.0        353.0       141.0
3     35   165.0    59.0          93.0        184.0        367.0       173.0
4     54   172.0    95.0         138.0        163.0        386.0       185.0
5     27   167.0    48.0          80.0        152.0        343.0       164.0
6     63   174.0    79.0          91.0        151.0        410.0       198.0
7     18   175.0    60.0         102.0        135.0        379.0       167.0
8     46   162.0    65.0          68.0        175.0        391.0       147.0
9     47   162.0    74.0         102.0        126.0        357.0       135.0
10    64   160.0    60.0          87.0        131.0        396.0       169.0
11    63   163.0    62.0          79.0        160.0        371.0       171.0
12    42   178.0    75.0          87.0        136.0        356.0       202.0
13    80   173.0    85.0          93.0        183.0        358.0       149.0
14    48   160.0    61.0          72.0        204.0        360.0       155.0
15    45   169.0    73.0          92.0        124.0        370.0       180.0
16    48   182.0    77.0          77.0        196.0        325.0       235.0
17    48   157.0    95.0          85.0        177.0        368.0       250.0
18    79   150.0    60.0          93.0        178.0        361.0       132.0
19    49   166.0    87.0          78.0          0.0        322.0       122.0
20    45   158.0    65.0          82.0        122.0        336.0       174.0
21    75   162.0    74.0          82.0        176.0        357.0       142.0
22    35   160.0    51.0          71.0        124.0        367.0       168.0
23    66   160.0    60.0          73.0          0.0        364.0       144.0
24    58   166.0    70.0          89.0        158.0        355.0       144.0
25    71   156.0    56.0          78.0        195.0        331.0       148.0
26    58   160.0    65.0         133.0        148.0        417.0       260.0
27    32   164.0    57.0          77.0        144.0        340.0       148.0
28    41   154.0    75.0          88.0        157.0        384.0       132.0
29    40   160.0    75.0          79.0        154.0        350.0       160.0
30    70   160.0    90.0          76.0        187.0        396.0       155.0
31    72   160.0    70.0          79.0        152.0        368.0       156.0
```

Gambar 1.20

Melakukan folding pada data dengan membaginya menjadi tiga dan disimpan pada tiga variabel berbeda kemudian mengeluarkan outputnya.

BAB II

DASAR TEORI

2.1. DECISION TREE (DT)

Decision Tree adalah algoritma *machine learning* yang menggunakan seperangkat aturan untuk membuat keputusan dengan terstruktur seperti pohon yang memodelkan kemungkinan hasil, biaya sumber daya, utilitas dan kemungkinan konsekuensi atau resiko. Konsepnya adalah dengan cara menyajikan algoritma dengan pernyataan bersyarat, yang meliputi cabang untuk mewakili langkah-langkah pengambilan keputusan yang dapat mengarah pada hasil yang menguntungkan.

Decision Tree juga berguna untuk dieksplorasi data, menemukan hubungan antara sejumlah calon variabel input dengan sebuah variabel target. Pohon keputusan eksplorasi data dan pemodelan yang salah langkah pertama yang sangat baik dalam proses pemodelan yang digunakan sebagai model akhir untuk beberapa teknik lainnya.

2.2 K-Nearest Neighboor (KNN)

K-Nearest Neighbor termasuk salah satu algoritma paling sederhana yang digunakan dalam machine learning untuk regresi dan klasifikasi. KNN mengikuti strategi "*bird of a feather*" dalam menentukan di mana data baru sebaiknya ditempatkan. Algoritma KNN mengasumsikan bahwa sesuatu yang mirip akan ada dalam jarak yang berdekatan atau bertetangga. Artinya data-data yang cenderung serupa akan dekat satu sama lain.

Langkah pertama dalam algoritma KNN adalah mengukur jarak antara data baru yang akan diprediksi dengan setiap data pelatihan yang ada. Pada umumnya, jarak diukur menggunakan metrik Euclidean distance, tetapi tergantung pada konteks masalah, dapat juga menggunakan metrik jarak lain seperti Manhattan distance atau Minkowski distance. Setelah mengukur jarak, langkah selanjutnya adalah menentukan nilai K, yaitu jumlah tetangga terdekat yang akan digunakan dalam proses klasifikasi atau regresi. Pemilihan nilai K ini sangat penting, karena dapat mempengaruhi hasil prediksi. Nilai K yang terlalu kecil dapat menyebabkan model terlalu sensitif terhadap noise, sementara nilai K yang terlalu besar dapat menyebabkan kehilangan kemampuan untuk menangkap pola yang kompleks.

Import Library

```
import pandas as pd
import numpy as np
import math
import matplotlib
from matplotlib import pyplot as plt
from statsmodels.graphics.gofplots import qqplot
import csv
```

BAB III
EVALUASI HASIL DAN DISKUSI

Metode KNN menghasilkan hasil yang cukup baik dalam melakukan klasifikasi atau regresi. Salah satu keunggulan KNN adalah fleksibilitasnya dalam menangani data dengan fitur kompleks atau non-linear. Namun, dalam penggunaannya, penting untuk memilih nilai K yang optimal. Nilai K yang terlalu kecil dapat menyebabkan model terlalu sensitif terhadap noise dan menghasilkan hasil yang tidak stabil, sementara nilai K yang terlalu besar dapat menyebabkan kehilangan kemampuan untuk menangkap pola yang kompleks. Oleh karena itu, penentuan nilai K yang tepat melalui eksperimen dan evaluasi menjadi kunci penting dalam menggunakan metode KNN.

Metode Decision Tree juga memberikan hasil yang lebih baik dalam pemodelan prediktif. Kelebihan utama DT adalah kemampuannya menghasilkan model yang mudah dipahami dan diinterpretasikan. Namun, DT rentan terhadap overfitting jika tidak dikendalikan dengan baik. Overfitting terjadi ketika model terlalu kompleks dan secara tidak akurat mempelajari pola pada data latih, sehingga kinerjanya menurun pada data uji atau data baru. Oleh karena itu, pemangkasan (pruning) menjadi langkah penting dalam mengurangi kompleksitas model dan mencegah overfitting. Selain itu, penyesuaian parameter seperti kedalaman maksimum pohon (max_depth) dan jumlah sampel minimum di setiap daun (min_samples_leaf) juga dapat membantu mengoptimalkan model sesuai dengan data dan tujuan analisis.

Pengukuran performansi pada kedua metode (KNN dan DT) dilakukan dengan menggunakan metrik evaluasi yang sesuai, seperti akurasi, presisi, dan recall. Hasil dan pengukuran performansi dari KNN adalah 0.56 dan DT adalah 77 %. Evaluasi ini dilakukan untuk menilai seberapa baik kedua metode dapat melakukan prediksi dan mengklasifikasikan data dengan benar. Selama proses pengukuran performansi, perlu diperhatikan bahwa hasil evaluasi dapat bervariasi tergantung pada dataset yang digunakan, pengaturan parameter, dan teknik preprocessing data yang diterapkan. Oleh karena itu, perlu melakukan validasi silang (cross-validation) atau pemisahan dataset menjadi data latih (training data) dan data uji (test data) untuk mendapatkan hasil yang lebih akurat dan reliabel.

BAB IV
KESIMPULAN DAN SARAN

4.1 Kesimpulan

Dari analisis yang dilakukan, dapat disimpulkan bahwa algoritma Decision Tree (DT) lebih baik dari algoritma K-Nearest Neighbors (kNN) dalam permasalahan ini. Decision Tree (DT) mampu menghasilkan model yang mudah dipahami dan dapat diinterpretasikan dengan baik. Di sisi lain, algoritma K-Nearest Neighbors (kNN) memiliki fleksibilitas yang tinggi dalam menangani data dengan fitur kompleks atau non-linear. Namun, pemilihan nilai K yang tepat sangat krusial untuk menghindari sensitivitas yang berlebihan terhadap noise.

4.2 Saran

Untuk algoritma Decision Tree (DT), beberapa saran dapat diterapkan agar hasilnya lebih optimal. Pertama, penting untuk melakukan pemangkasan (pruning) pada pohon keputusan untuk mengurangi kompleksitas model dan mencegah overfitting. Dengan menghapus cabang-cabang yang tidak signifikan, model akan menjadi lebih sederhana namun tetap mempertahankan kemampuan prediktifnya. Selanjutnya, penyesuaian parameter seperti kedalaman maksimum pohon (max_depth) dan jumlah sampel minimum di setiap daun (min_samples_leaf) dapat membantu mengoptimalkan model sesuai dengan data dan tujuan analisis. Jika terjadi perubahan besar pada data pelatihan, disarankan untuk memperbarui atau melatih ulang model DT agar tetap relevan dengan data baru.

BAB V
DAFTAR PUSTAKA

[1] *Ayuni Qurrata, Randy Cahya Wihandika, Novanto Yudistira, Klasifikasi Aritmia Dari Hasil Elektrokardiogram Menggunakan Metode Support Vector Machine, 2021.*

[2] *Ramadhan Ardi Satyo, Decision Tree Algoritma Beserta Contohnya Pada Data Mining, 2022.*

[3] *lp2m, Algoritma K-Nearest Neighbors (KNN)-Pengertian dan Penerapan, 2023.*

link source code and others:
https://drive.google.com/drive/folders/1hKt8xL4nzQ59dFeCRpfHyyoXxT05AoUo?usp=sharing