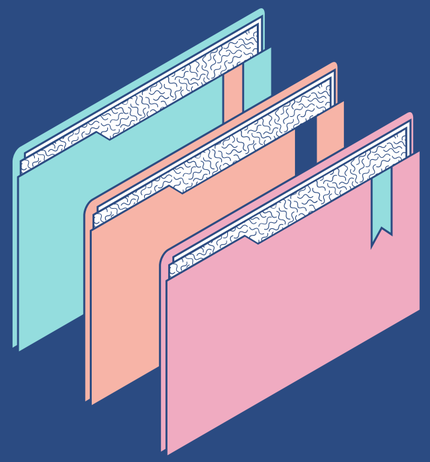
PENGANTAR KECERDASAN BUATAN

Analisis Data Arrhythmia dengan Algoritma Learning (kNN dan Naive Bayes)

Disusun Oleh :

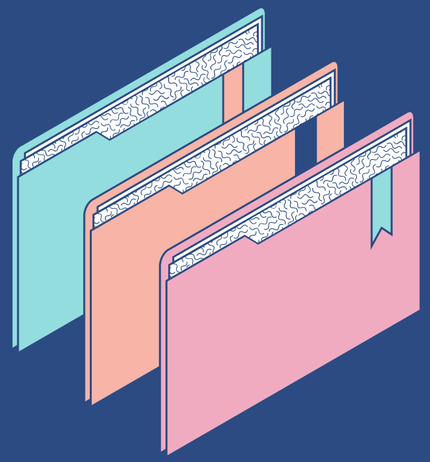
Annisa Izzatul Latifa	1301213328
Firman Hoerulloh	1301213392
Reza Mu'ammar Widyanto	1301210513
Renaldhy Vebryan Hermanto	1301213335



Pendahuluan

ARRHYTHMIA DATASET

INFORMASI MEDIS MENGENAI PASIEN DIDIAGNOSA MEMILIKI PENYAKIT ARRHYTHMIA YAITU GANGGUAN PADA DETAK JANTUNG, MEMBEDAKAN ADA TIADANYA ARITMIA JANTUNG DAN MENGKLASIFIKASIKANNYA KE DALAM SALAH SATU DARI ENAM BELAS KELAS.



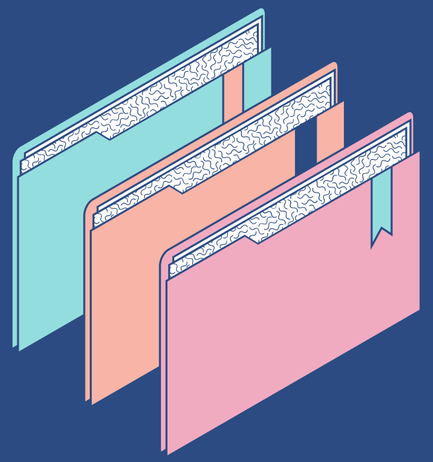
Pendahuluan

ARRHYTHMIA DATASET

- JUMLAH SEMPEL DAN ATRIBUT:
452 SAMPEL DAN 280 ATRIBUT

- TIPE DATA :
KATEGORIKAL DAN NUMERIK

- KOLOM TARGET :
TERDAPAT KELAS ELEKTOKARDIOGRAF 1 -16 KELAS



ALGORITMA KNN (K-NEAREST NEIGHBOR)

dengan mengidentifikasi tetangga terdekat dari titik kueri yang diberikan

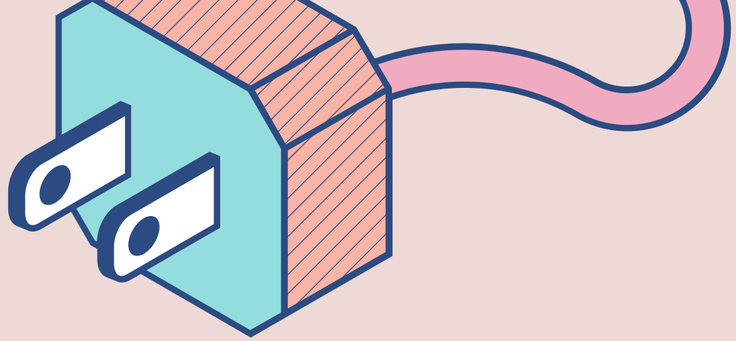
algoritma mechine learning yang dapat digunakan untuk dataset dengan keberagam fitur, karena dataset arrhythmia memiliki banyak fitur seperti tekanan darah, denyut nadi, dan jenis aritmia

ALGORITMA NAIVE BAYES

metode klasifikasi data berdasarkan probabilitas kelas berdasarkan fitur-fitur dan memilih dengan probabilitas tertinggi.

naive bayes dapat memperoleh informasi dari atribut kategorikal seperti jenis aritmia

Pra-pemrosesan data



a) library

```
import urllib.request
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import urllib.request
import pandas as pd
import numpy as np
import math
```

b) Membaca dataset dan menambah kolom

```
# melakukan import dataset dari URL
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/arrhythmia/arrhythmia.data'
filename = 'arrhythmia.data'

urllib.request.urlretrieve(url, filename)
```

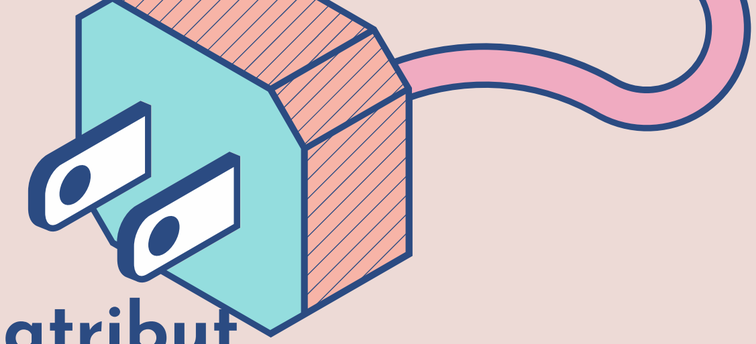
```
('arrhythmia.data', <http.client.HTTPMessage at 0x7ff971a668c0>)
```

```
# Membaca dataset
df = pd.read_csv('arrhythmia.data', header=None)

column_names = [
    'age','gender','height','weight','qrs_duration','p-r_Interval','q-t_interval','t_interval','p_interval',
            'QRS','T','P','QRST','J','heart_rate','Q_D1','R_D1','S_D1','R\'_D1','S\'_D1','NOD_D1','EOR_R_D1','EOD_R_D1',
            'EOR_P_D1','EOD_P_D1','EOR_T_D1','EOD_T_D1','Q_D2','R_D2','S_D2','R\'_D2','S\'_D2','NOD_D2','EOR_R_D2','EOD_R_D2',
            'EOR_P_D2','EOD_P_D2','EOR_T_D2','EOD_T_D2','Q_D3','R_D3','S_D3','R\'_D3','S\'_D3','NOD_D3','EOR_R_D3','EOD_R_D3',
            'EOR_P_D3','EOD_P_D3','EOR_T_D3','EOD_T_D3','Q_AVR','R_AVR','S_AVR','R\'_AVR','S\'_AVR','NOD_AVR','EOR_R_AVR',
            'EOD_R_AVR','EOR_P_AVR','EOD_P_AVR','EOR_T_AVR','EOD_T_AVR','Q_AVL','R_AVL','S_AVL','R\'_AVL','S\'_AVL','NOD_AVL',
            'EOR_R_AVL','EOD_R_AVL','EOR_P_AVL','EOD_P_AVL','EOR_T_AVL','EOD_T_AVL','Q_AVF','R_AVF','S_AVF','R\'_AVF','S\'_AVF',
            'NOD_AVF','EOR_R_AVF','EOD_R_AVF','EOR_P_AVF','EOD_P_AVF','EOR_T_AVF','EOD_T_AVF','Q_V1','R_V1','S_V1','R\'_V1',
            'S\'_V1','NOD_V1','EOR_R_V1','EOD_R_V1','EOR_P_V1','EOD_P_V1','EOR_T_V1','EOD_T_V1','Q_V2','R_V2','S_V2','R\'_V2',
            'S\'_V2','NOD_V2','EOR_R_V2','EOD_R_V2','EOR_P_V2','EOD_P_V2','EOR_T_V2','EOD_T_V2','Q_V3','R_V3','S_V3','R\'_V3',
            'S\'_V3','NOD_V3','EOR_R_V3','EOD_R_V3','EOR_P_V3','EOD_P_V3','EOR_T_V3','EOD_T_V3','Q_V4','R_V4','S_V4','R\'_V4',
            'S\'_V4','NOD_V4','EOR_R_V4','EOD_R_V4','EOR_P_V4','EOD_P_V4','EOR_T_V4','EOD_T_V4','Q_V5','R_V5','S_V5','R\'_V5',
            'S\'_V5','NOD_V5','EOR_R_V5','EOD_R_V5','EOR_P_V5','EOD_P_V5','EOR_T_V5','EOD_T_V5','Q_V6','R_V6','S_V6','R\'_V6',
            'S\'_V6','NOD_V6','EOR_R_V6','EOD_R_V6','EOR_P_V6','EOD_P_V6','EOR_T_V6','EOD_T_V6','A_JJ_D1','A_Q_D1','A_R_D1',
            'A_S_D1','A_R\'_D1','A_S\'_D1','A_P_D1','A_T_D1','QRSA_D1','QRSTA_D1','A_JJ_D2','A_Q_D2','A_R_D2','A_S_D2',
            'A_R\'_D2','A_S\'_D2','A_P_D2','A_T_D2','QRSA_D2','QRSTA_D2','A_JJ_D3','A_Q_D3','A_R_D3','A_S_D3','A_R\'_D3',
            'A_S\'_D3','A_P_D3','A_T_D3','QRSA_D3','QRSTA_D3','A_JJ_AVR','A_Q_AVR','A_R_AVR','A_S_AVR','A_R\'_AVR','A_S\'_AVR',
            'A_P_AVR','A_T_AVR','QRSA_AVR','QRSTA_AVR','A_JJ_AVL','A_Q_AVL','A_R_AVL','A_S_AVL','A_R\'_AVL','A_S\'_AVL','A_P_AVL',
            'A_T_AVL','QRSA_AVL','QRSTA_AVL','A_JJ_AVF','A_Q_AVF','A_R_AVF','A_S_AVF','A_R\'_AVF','A_S\'_AVF','A_P_AVF','A_T_AVF',
            'QRSA_AVF','QRSTA_AVF','A_JJ_V1','A_Q_V1','A_R_V1','A_S_V1','A_R\'_V1','A_S\'_V1','A_P_V1','A_T_V1','QRSA_V1','QRSTA_V1',
            'A_JJ_V2','A_Q_V2','A_R_V2','A_S_V2','A_R\'_V2','A_S\'_V2','A_P_V2','A_T_V2','QRSA_V2','QRSTA_V2','A_JJ_V3','A_Q_V3',
            'A_R_V3','A_S_V3','A_R\'_V3','A_S\'_V3','A_P_V3','A_T_V3','QRSA_V1','QRSTA_V3','A_JJ_V4','A_Q_V4','A_R_V4','A_S_V4',
            'A_R\'_V4','A_S\'_V4','A_P_V4','A_T_V4','QRSA_V4','QRSTA_V4','A_JJ_V5','A_Q_V5','A_R_V5','A_S_V5','A_R\'_V5','A_S\'_V5',
            'A_P_V5','A_T_V5','QRSA_V5','QRSTA_V5','A_JJ_V6','A_Q_V6','A_R_V6','A_S_V6','A_R\'_V6','A_S\'_V6','A_P_V6','A_T_V6',
            'QRSA_V6','QRSTA_V6','Class'
]

df.columns = column_names
```

Pra-pemrosesan data



c) pada penelitian ini kami menggunakan semua atribut

```
df.head()  # Menampilkan beberapa baris pertama dataset
```

	age	gender	height	weight	qrs_duration	p-r_Interval	q-t_interval	t_interval	p_interval	QRS	...	A_Q_V6	A_R_V6	A_S_V6	A_R'_V6	A_S'_V6	A_P_V6	A_T_V6	QRSA_V6	QRSTA_V6	Class
0	75	0	190	80	91	193	371	174	121	-16	...	0.0	9.0	-0.9	0.0	0.0	0.9	2.9	23.3	49.4	8
1	56	1	165	64	81	174	401	149	39	25	...	0.0	8.5	0.0	0.0	0.0	0.2	2.1	20.4	38.8	6
2	54	0	172	95	138	163	386	185	102	96	...	0.0	9.5	-2.4	0.0	0.0	0.3	3.4	12.3	49.0	10
3	55	0	175	94	100	202	380	179	143	28	...	0.0	12.2	-2.2	0.0	0.0	0.4	2.6	34.6	61.6	1
4	75	0	190	80	88	181	360	177	103	-16	...	0.0	13.1	-3.6	0.0	0.0	-0.1	3.9	25.4	62.8	7

A

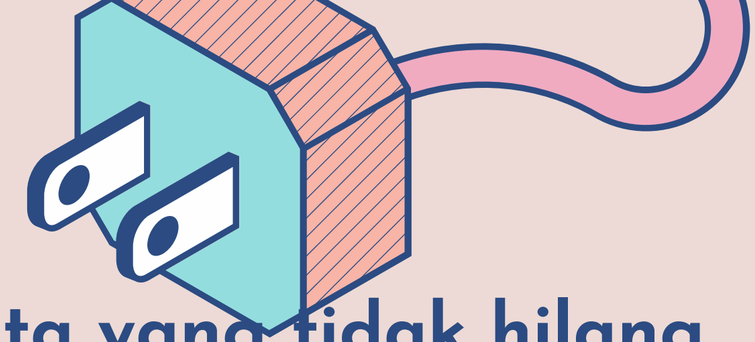
d) Memeriksa Missing Value

```
# melakukan pemeriksaan data yang tidak lengkap
missing_values = dict()
for column in df:
    for data in df[column]:
        if data == '?':
            missing_values[column] = missing_values.get(column, 0) + 1

print("Jumlah data yang hilang:")
for keys, values in missing_values.items():
  print(keys, ":", values)
```

```
Jumlah data yang hilang:
T : 8
P : 22
QRST : 1
J : 376
heart_rate : 1
```

Pra-pemrosesan data



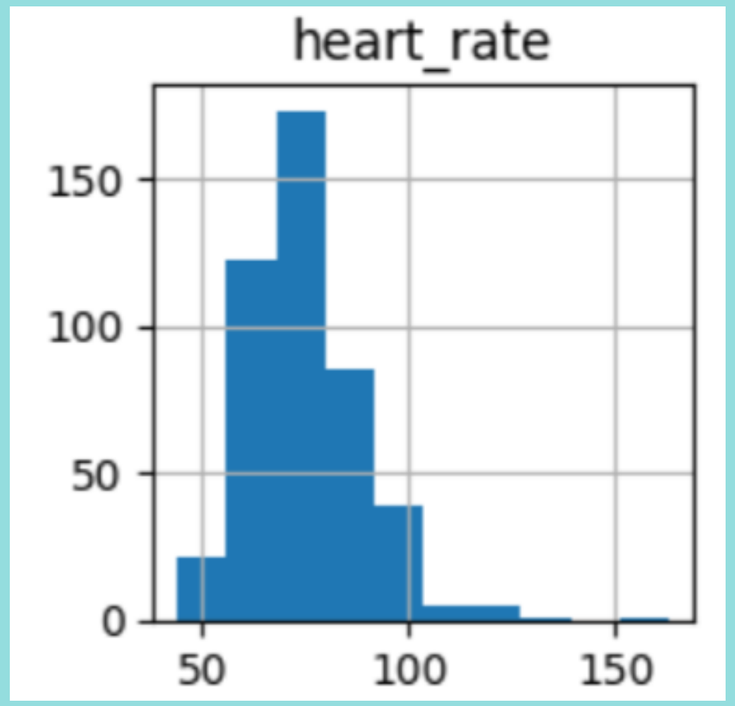
e) mengisi missing value dengan rata-rata dari data yang tidak hilang

```python
# menghapus baris dengan data yang hilang
def data_clean(df):
  # dari pemeriksaan data di atas, didapat bahwa kolom 'heart_rate', 'T', 'P', 'QRST', dan 'J' memiliki data yang hilang
  for i in ['heart_rate', 'T', 'P', 'QRST', 'J']:
    df[i] = df[i].replace('?','-1').astype(int)
    mean = df[i].mean(axis=0)
    df[i] = df[i].replace(-1,mean)
  return df
```

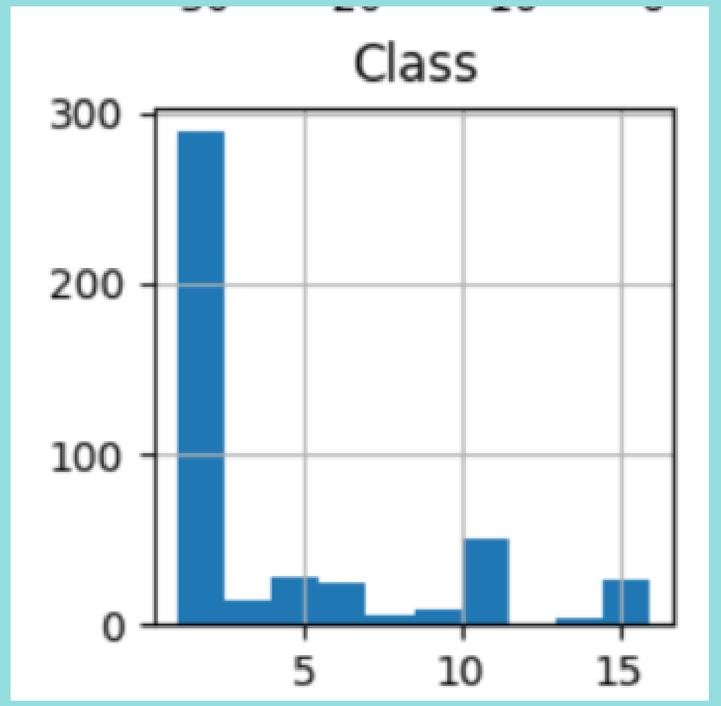
f) deskripsi statistik

```python
df.describe() # Menampilkan ringkasan statistik deskriptif untuk setiap kolom numerik
```

	age	gender	height	weight	qrs_duration	p-r_Interval	q-t_interval	t_interval	p_interval	QRS	...	A_Q_V6	A_R_V6	A_S_V6	A_R'_V6	A_S'_V6	A_P_V6	A_T_V6	QRSA_V6	QRSTA_V6	Class
count	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	...	452.000000	452.000000	452.000000	452.000000	452.0	452.000000	452.000000	452.000000	452.000000	452.000000
mean	46.471239	0.550885	166.188053	68.170354	88.920354	155.152655	367.207965	169.949115	90.004425	33.676991	...	-0.278982	9.048009	-1.457301	0.003982	0.0	0.514823	1.222345	19.326106	29.473230	3.880531
std	16.466631	0.497955	37.170340	16.590803	15.364394	44.842283	33.385421	35.633072	25.826643	45.431434	...	0.548876	3.472862	2.002430	0.050118	0.0	0.347531	1.426052	13.503922	18.493927	4.407097
min	0.000000	0.000000	105.000000	6.000000	55.000000	0.000000	232.000000	108.000000	0.000000	-172.000000	...	-4.100000	0.000000	-28.600000	0.000000	0.0	-0.800000	-6.000000	-44.200000	-38.600000	1.000000
25%	36.000000	0.000000	160.000000	59.000000	80.000000	142.000000	350.000000	148.000000	79.000000	3.750000	...	-0.425000	6.600000	-2.100000	0.000000	0.0	0.400000	0.500000	11.450000	17.550000	1.000000
50%	47.000000	1.000000	164.000000	68.000000	86.000000	157.000000	367.000000	162.000000	91.000000	40.000000	...	0.000000	8.800000	-1.100000	0.000000	0.0	0.500000	1.350000	18.100000	27.900000	1.000000
75%	58.000000	1.000000	170.000000	79.000000	94.000000	175.000000	384.000000	179.000000	102.000000	66.000000	...	0.000000	11.200000	0.000000	0.000000	0.0	0.700000	2.100000	25.825000	41.125000	6.000000
max	83.000000	1.000000	780.000000	176.000000	188.000000	524.000000	509.000000	381.000000	205.000000	169.000000	...	0.000000	23.600000	0.000000	0.800000	0.0	2.400000	6.000000	88.800000	115.900000	16.000000



DARI 451 DATA TERLIHAT HEART RATE PALING BANYAK PADA >60 DAN < 80 BPM DENGAN JUMLAH LEBIH DARI 150 ORANG



DARI 451 DATA TERLIHAT ARRHTHMIA CLASS PALING BANYAK BERADA PADA KELAS NORMAL DENGAN JUMLAH LEBIH DARI 200

Dasar Teori

Algoritma learning kNN (k-Nearest Neighbors) adalah algoritma paling sederhana yang merupakan sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran (train datasets) yang jaraknya paling dekat dengan objek tersebut.

Teknik pencarian tetangga terdekat yang umum dilakukan dengan menggunakan formula jarak euclidean

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Algoritma Naive Bayes merupakan metode yang cocok untuk klasifikasi biner dan multi class dengan variabel input kategori daripada numerik yang menggunakan probabilitas bersyarat, Probabilitas bersyarat adalah ukuran peluang suatu peristiwa yang terjadi berdasarkan peristiwa lain yang telah terjadi.

Rumus Probabilitas Bersyarat

$$P(A|B) = \frac{P(B|A)*P(A)}{P(B)}$$

Cara Kerja Algoritma

k-Nearest Neighbor

1. Load dataset Arrhythmia ke dalam Python
2. Pisahkan fitur-fitur dan label dari dataset.
3. Lakukan fold pada data
4. Lakukan normalisasi atau standarisasi data jika diperlukan.
5. Bagi dataset menjadi data pelatihan dan data pengujian.
6. Hitung jarak antara data uji dengan setiap data pelatihan menggunakan matrik jarak seperti Euclidean distance.
7. Pilih k tetangga terdekat.
8. Gunakan mayoritas kelas dari tetangga terdekat sebagai prediksi kelas untuk data uji.
9. Evaluasi performa algoritma menggunakan matrik evaluasi seperti akurasi.

Naive Bayes

1. Load dataset Arrhythmia ke dalam Python
2. Pisahkan fitur-fitur dan label dari dataset.
3. Lakukan preprocessing data seperti normalisasi jika diperlukan.
4. Bagi dataset menjadi data pelatihan dan data pengujian.
5. Hitung Mean untuk setiap kelas dalam data pelatihan
6. Hitung probabilitas untuk setiap kelas dalam data pelatihan.
7. Evaluasi performa algoritma menggunakan matrik evaluasi seperti akurasi.

Kode Program

Untuk kode program yang telah kami buat dapat diakses pada link berikut :

kNN

https://github.com/muammarrz/tubes-ai/blob/3f21f217ff0c6e7291915875ab4852ac6e3b9949/arrhythmia_kNN.ipynb

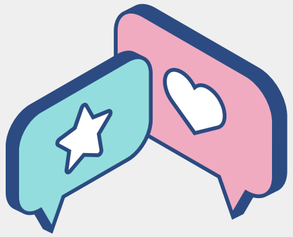
Naive Bayes

https://github.com/muammarrz/tubes-ai/blob/3f21f217ff0c6e7291915875ab4852ac6e3b9949/arrhythmia_Naive_Bayes.ipynb

Github

https://github.com/muammarrz/tubes-ai

Evaluasi Hasil dan Diskusi



a. Rumus Performansi

```
if(x):
    print(f"\nTP : {TP} FN : {FP}\nTN : {TN} FN : {FN}")
    print(f"Accuracy : {((TP+TN)/(TP+TN+FP+FN))*100}%")
    print(f"Precission : {((TP)/(TP+FP))*100}%")
    print(f"Recall : {((TP)/(TP+FN))*100}%")
```

Akurasi : (jumlah prediksi benar) / (jumlah total prediksi) * 100%

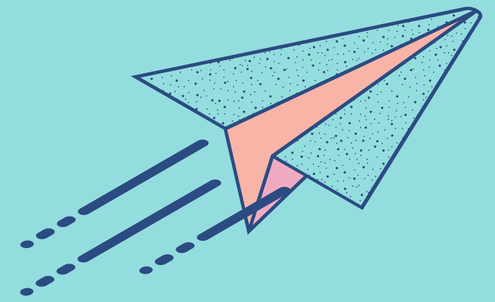
Presisi : (Jumlah True Positive) / (Jumlah True Positive + Jumlah False Positive) * 100%

Recall : (Jumlah True Positive) / (Jumlah True Positive + Jumlah False Negative) * 100%

F1 Score (penggabungan presisi dan recall) : 2 * ((Presisi * Recall) / (Presisi + Recall))

Hasil

Algoritma kNN



Jarak Euclidean :

PERHITUNGAN JARAK EUCLIDEAN

```
# Perhitungan jarak Euclidean
def euclidean(x1,x2):
  return np.sqrt(np.sum((x1 - x2)**2))

euclidean(x.iloc[0], x.iloc[1])
```

```
2.28570280218962
```

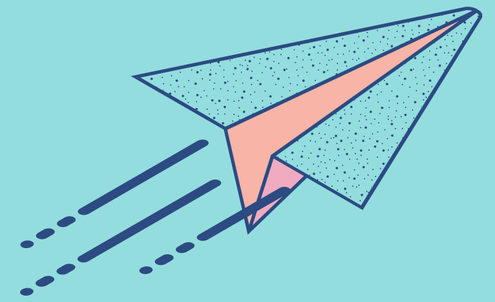
Nilai Akurasi :

```
[ ]  # Evaluasi kinerja model menggunakan akurasi
     k = 10
     accs = []
     folds = [fold1, fold2, fold1]
     for i in range(len(folds)):
       accs.append(evaluate(folds[i],k))
     print(f'Menggunakan k : {k}, dengan rata-rata akurasi : {sum(accs)/3}')
```

```
Menggunakan k : 10, dengan rata-rata akurasi : 0.5666666666666668
```

Hasil

Algoritma Naive Bayes



```
TP : 17 FN : 20
TN : 37 FN : 62
Accuracy : 39.7058823529411174%
Precission : 45.945945945945595%
Recall : 21.51898734177215 3%
```

Nilai Akurasi : 39.7058823529411174%

Nilai Presisi : 45.945945945945595%

Nilai Recall : 21.51898734177215 3%

Analisis

ALGORITMA KNN

- Dalam konteks kNN, nilai Euclidean digunakan untuk menentukan tetangga terdekat dari suatu data. Nilai Euclidean yang diperoleh, yaitu 2.285702802189 62, menunjukkan adanya kemiripan antara data yang diuji dengan data latihan terdekat.

- Nilai akurasi sebesar 56.666666666666 8% menunjukkan bahwa algoritma kNN dapat melakukan klasifikasi dengan tingkat keakuratan sebesar 56.66%. Akurasi ini menggambarkan seberapa baik algoritma dapat memprediksi kelas yang benar.

ALGORITMA NAIVE BAYES

- Nilai akurasi sebesar 39.705882352941174% menunjukkan bahwa algoritma Naive Bayes memiliki tingkat keakuratan sebesar 40.44% dalam melakukan klasifikasi pada dataset tersebut.

- Nilai presisi sebesar 45.945945945945 95% mengindikasikan seberapa baik algoritma dapat mengidentifikasi kelas positif dengan benar dari keseluruhan prediksi yang dilakukan.

- Nilai recall sebesar 21.518987341772153% menggambarkan kemampuan algoritma dalam mengidentifikasi dan mendeteksi data yang benar dari kelas positif.

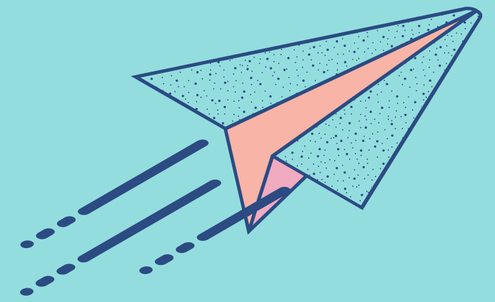
Perbandingan

Berdasarkan nilai akurasi yang diperoleh, algoritma kNN memiliki akurasi yang lebih baik dibandingkan dengan Naive Bayes. Algoritma kNN mencapai akurasi sebesar 56.66%, sedangkan Naive Bayes mencapai akurasi sebesar 39.70%. Oleh karena itu, jika tujuan utama adalah mencapai akurasi yang lebih tinggi, algoritma kNN dapat dianggap lebih baik dalam konteks ini.
Berdasarkan Confusion Matrix pada algoritma Naive Bayes terdapat Confusion Matrix yaitu

TP : 17 FN : 20
TN : 37 FN : 62
Accuracy : 39.705882352941174%
Precission : 45.945945945945%
Recall : 21.518987341772153%

tetapi karena pada algoritma kNN tidak terdapat Confusion Matrix, tidak perlu adanya perbandingan.

KESIMPULAN



kNN *(k-Nearest Neighbors)* memiliki akurasi yang lebih tinggi daripada *Naive Bayes*, jadi algoritma yang lebih baik dalam penelitian kami untuk *Arrhythmia Dataset* adalah algoritma kNN *(k-Nearest Neighbors)*
kinerja antara dua algoritma tersebut dapat berbeda-beda sesuai dengan kondisi algoritma dan data.

SEKIAN PRESENTASI KAMI

APAKAH ADA PERTANYAAN?