Laporan Tugas Besar

Mata Kuliah Pengantar Kecerdasan Buatan

Disusun Oleh:

Muhammad Hermawan Alghozy	**(1301213473)**
Dito Rifadli Febrian	**(1301213518)**
Raditha Ariyani	**(1301213527)**

Kelompok 10

Kelas : IF-45-12

Universitas Telkom

2023

BAB I

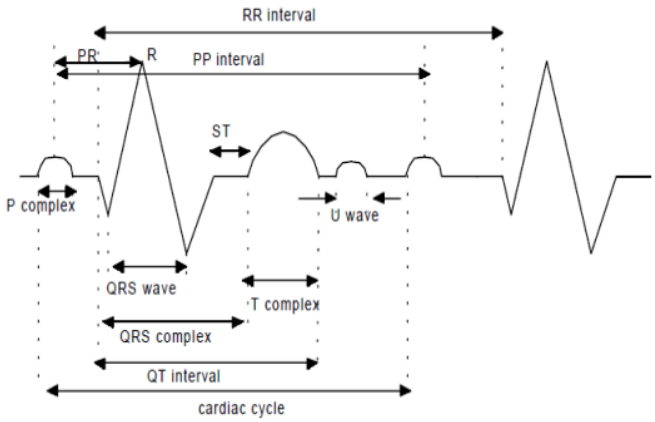
PENDAHULUAN

1.1 Latar Belakang

Terdapat sebuah penyakit jantung yang bernama Arrythmia. Arrhythmia adalah penyakit gangguan pada detak jantung yang tidak teratur, bisa terlalu cepat atau lambat.Sehingga perlu suatu data untuk dianalisis. Dalam mempelajari data tersebut terdapat challenging untuk mengolahnya, yakni dengan cara menerapkan teknik pra-pemrosesan data melalui pemahaman mengenai EDA (Exploratory Data Analytics). Banyak penelitian dilakukan dalam beberapa tahun terakhir mengenai klasifikasi aritmia. HA Guvenir menciptakan algoritma baru yang disebut VF15 untuk mengklasifikasikan aritmia. Dia menunjukkan bahwa algoritma baru ini berperforma sangat baik dibandingkan dengan algoritma klasifikasi dasar.Terdapat banyak penelitian yang dilakukan di bidang ini dan masih terdapat kemungkinan untuk perbaikan. Dalam makalah ini, kami telah membandingkan dua algoritma klasifikasi, yaitu knn dan Naïve Bayes untuk menentukan apakah pasien menderita aritmia atau tidak.

Laporan ini akan membahas tentang teknik pra-pemrosesan data melalui pemahaman mengenai EDA (Exploratory Data Analytics). EDA merupakan suatu pendekatan analisis data yang digunakan untuk menggali wawasan dan pemahaman awal terhadap data yang akan diproses. Dalam konteks ini, fokus utama laporan adalah mengenai teknik dan langkah-langkah yang dilakukan dalam melakukan EDA untuk mempersiapkan data sebelum masuk ke tahap pemrosesan lebih lanjut. Pemahaman yang baik tentang EDA sangat penting karena dapat membantu dalam mengidentifikasi kelemahan, kesalahan, atau anomali dalam data yang akan mempengaruhi hasil pemrosesan. Dengan melakukan EDA dengan baik, kita dapat menemukan informasi penting, tren, pola, atau hubungan antar variabel yang akan memberikan wawasan yang lebih baik dalam mengambil keputusan yang didukung oleh data.

Dalam laporan ini, kami akan menggunakan teknik EDA menggunakan algoritma kNN dan *Naive Bayes*. Tujuan dari laporan ini adalah untuk memberikan pemahaman yang lebih baik tentang pentingnya EDA dan juga agar kita dapat menemukan hasil data dengan menggunakan algoritma yang kita pilih. Selanjutnya, laporan ini akan menguraikan teknik-teknik algoritma EDA yang relevan yang dapat diterapkan dalam pra-pemrosesan data.



Gambar 1. Siklus Jantung dari ECG

1.2 Deskripsi Dataset

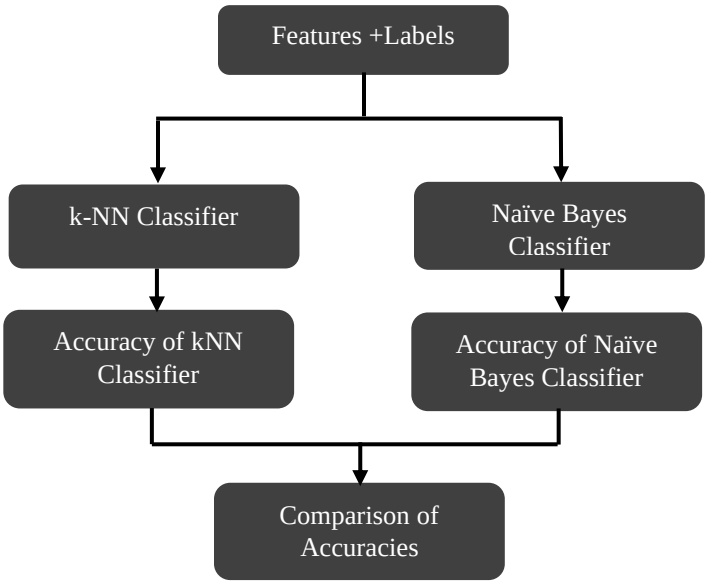
Data ini berisi 279 atribut, 206 di antaranya adalah linier dan sisanya nominal. Dalam dataset berisi catatan 452 pasien dan 279 nilai fitur. Tujuannya untuk membedakan apakah ada atau tidak adanya *aritmia* jantung dan untuk mengklasifikasikannya dalam salah satu dari 16 kelompok. Kelas 01 mengacu pada 'normal' Kelas ECG 02 hingga 15 mengacu pada kelas *aritmia* yang berbeda dan kelas 16 mengacu pada sisa yang tidak terklasifikasi. Kelas-kelas dan jenis *aritmia* dijelaskan pada Tabel I.

Class	Description
1	Normal
2	Ischemic changes (Coronary Artery Disease)
3	Old Anterior Myocardial Infarction
4	Old Inferior Myocardial Infarction
5	Sinus tachycardy
6	Sinus bradycardy
7	Ventricular Premature Contraction (PVC)
8	Supraventricular Premature Contraction
9	Left bundle branch block
10	Right bundle branch block
11	1. degree AtrioVentricular block
12	2. degree AV block
13	3. degree AV block
14	Left ventricule hypertrophy
15	Atrial Fibrillation or Flutter
16	Others

Tabel 2. Class Distribution of Arrhythmia

1.3 Model Sistem

Model sistem yang digunakan untuk klasifikasi data menggunakan dua algortima yang berbeda yaitu: k-Nearest Neighbor dan Naïve Bayes.



Gambar 2. Model Sistem

Model sistem akan melewati beberapa tahap, yang pertama adalah menerima data set. Setelah itu, dilakukan klasifikasi menggunakan kNN atau Naïve Bayes. Setelah itu, nilai dari klasifikasi diakurasikan menjadi nilai data yang akurat, sehingga yang terakhir kita berhasil mendapatkan komparasi nilai data yang akurat.

1.4 Kontribusi Kelompok

Nama	Kontribusi
Muhammad Hermawan Alghozy	Pendahuluan, Kesimpulan
Dito Rifadlli Febrian	Dasar Teori
Raditha Ariyani	Evaluasi Hasil dan Diskui

“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”.

BAB II

DASAR TEORI

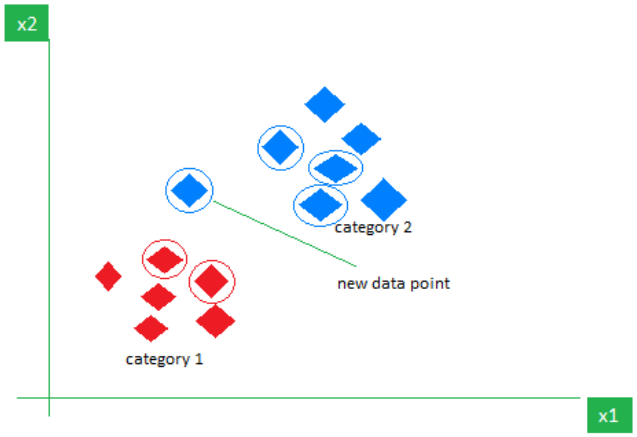
2.1 *K-Nearest Neighbor* (KNN)

A. Pengertian

K-Nearest Neighbor (KNN) adalah sebuah metode supervised yang berarti membutuhkan data training untuk mengklasifikasikan objek yang jaraknya paling dekat. Prinsip kerja *K-Nearest Neighbor* adalah mencari jarak terdekat antara data yang akan di evaluasi dengan *k* tetangga (*neighbor*) dalam data pelatihan (Whidhiasih et al., 2013).

B. Cara Kerja

Sebagai contoh, perhatikan tabel poin data berikut ini yang berisi dua fitur:



Gambar 3. Contoh cara kerja Algoritgma k-NN

Jika kita memplot titik-titik ini pada grafik, kita mungkin dapat menemukan beberapa klaster atau kelompok. Sekarang, dengan diberikan sebuah titik yang belum diklasifikasikan, kita dapat menempatkannya ke dalam sebuah kelompok dengan mengamati kelompok mana yang menjadi tetangga terdekatnya. Ini berarti sebuah titik yang dekat dengan sekelompok titik yang diklasifikasikan sebagai 'Merah' memiliki probabilitas yang lebih tinggi untuk diklasifikasikan sebagai 'Merah'.

Secara intuitif, kita dapat melihat bahwa titik pertama (2,5, 7) harus diklasifikasikan sebagai 'Hijau' dan titik kedua (5,5, 4,5) harus diklasifikasikan sebagai 'Merah'.

Jadi algoritma *k-nearest neighbor* (k-NN) digunakan untuk memberi label pada objek berdasarkan kriteria jarak minimum dari sampel pelatihan yang ada dalam vektor fitur. Dalam semua algoritma pembelajaran mesin, ini adalah algoritma paling sederhana. Kelas yang paling umum di antara tetangga terdekat k digunakan untuk memberi label pada objek, di mana k adalah

bilangan bulat positif. Tetangga dipilih dari himpunan objek yang memiliki label kelas yang sudah diketahui. Ini akan menjadi himpunan data pelatihan untuk algoritma. Sampel pelatihan berupa vektor dalam ruang fitur multidimensional, masing-masing dengan label kelas. Vektor fitur dan informasi kelas disimpan dalam sesi pelatihan algoritma. Bagian pengujian terdiri dari langkah-langkah berikut ini:

- Mengukur jarak Euclidean ke setiap titik pelatihan.
- Mencari k titik terdekat.
- Mengidentifikasi kelas yang paling umum di antara k titik terdekat.
- Menetapkan kelas tersebut.

Pemilihan nilai "k" ditentukan melalui data. Kita harus mengatur k sesuai dengan kebutuhan kita, misalnya jika ada noise dalam data, dipilih nilai k yang lebih besar untuk meminimalkan efek noise pada klasifikasi. Namun, karena nilai "k" yang tinggi, batas antara kelas-kelas menjadi kurang jelas. Jika nilai k diatur sebagai 1, ini merupakan kasus khusus di mana pengklasifikasi disebut sebagai pengklasifikasi tetangga terdekat. Jika fitur-fitur tidak konsisten dengan signifikansinya, akurasi algoritma k-NN dapat sangat terpengaruh oleh keberadaan fitur-fitur berisik atau tidak berhubungan.

Metrik Jarak yang Digunakan dalam Algoritma KNN:

- Euclidean Distance

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Manhattan Distance

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- Minkowski Distance

$$d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{\frac{1}{p}}$$

C. Implementasi Code

```
#Mengimpor library
import pandas as pd
import requests
import numpy as np
import math
```

```
#membaca dataset dari URL ke dalam DataFrame
```

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/arrhythmia/arrhythmia.data'
df = pd.read_csv(url, header=None)

df.head()  # Menampilkan beberapa baris pertama dari DataFrame

# Mengganti beberapa nama kolom dalam DataFrame dengan menggunakan dictionary
df = df.rename(columns={0: 'Age',1: 'Gender',2: 'Height',3: 'Weight',4:
'QRS_duration',5: 'P-R_interval',6: 'Q-T_interval', 7: 'T_interval', 8:
'P_interval',
                        9: 'QRS', 10: 'T', 11:'P', 12: 'QRST', 13: 'J', 14:
'Heart rate'})

df.QRS_duration.unique() # Menampilkan nilai unik dari kolom 'QRS_duration'

df.info() # Menampilkan informasi tentang DataFrame, termasuk jumlah entri
dan tipe data kolom

df.describe()  # Menampilkan ringkasan statistik dari DataFrame

df.shape # Menampilkan dimensi DataFrame (jumlah baris, jumlah kolom)

fold1 = (df.iloc[0:50].reset_index(drop=True),
df.iloc[50:150].reset_index(drop=True))
fold2 = (df.iloc[50:100].reset_index(drop=True), pd.concat([df.iloc[0:50],
df.iloc[100:150]]). reset_index(drop=True))
fold3 = (df.iloc[100:150].reset_index(drop=True),
df.iloc[0:100].reset_index(drop=True))

test, train = fold2
print(train) # Menampilkan data latih (train) dari lipatan kedua (fold 2)

# Fungsi untuk melakukan normalisasi pada DataFrame
def norm(df):
  df = df.replace('?', int(0))
  df = df.astype(int)
  df = (df - df.min()) / (df.max() - df.min())
  return df

x = df.drop('Height', axis=1)

x = norm(x)
x

# Fungsi untuk menghitung jarak Euclidean antara dua vektor
def euclidean(x1, x2):
```

```
  return np.sqrt(np.sum((x1 - x2)**2))

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

```
# Fungsi k-NN untuk melakukan klasifikasi
def knn(x_train, y_train, x_test, k):
  dist = []
  for row in range(x_train.shape[0]):
    dist.append(euclidean(x_train.iloc[row], x_test))

  data = x_train.copy()
  data['Dist'] = dist
  data['Class'] = y_train
  data = data.sort_values(by = 'Dist').reset_index(drop=True)

  y_pred = data.iloc[:k].Class.mode()
  return y_pred[0]
```

```
# Fungsi untuk menghitung akurasi prediksi
def acc(y_pred, y_true):
  true = 0
  for i in range(len(y_pred)):
    if y_pred[i] == y_true[i]:
      true+=1
  return true/len(y_pred)
```

```
# Fungsi untuk mengevaluasi performa model menggunakan metode validasi silang
def evaluate(fold, k):
  test, train = fold
  x_train, y_train = train.drop('Height', axis=1), train.Age
  x_test, y_test = test.drop('Height', axis=1), test.Age
  x_train = norm(x_train)
  x_test = norm(x_test)
  y_pred = []
  for row in range(x_test.shape[0]):
    y_pred.append(knn(x_train, y_train, x_test.iloc[row], k))

  return(acc(y_pred, y_test))
k = 5
accs = []
folds = [fold1, fold2, fold3]
for i in range(len(folds)):
  accs.append(evaluate(folds[1],k))

print(f'Menggunakan k: {k}, dengan rata-rata akurasi: {sum(accs)/3}')
```

Output Akhir:

```
[22] k = 5
     accs = []
     folds = [fold1, fold2, fold3]
     for i in range(len(folds)):
       accs.append(evaluate(folds[1],k))

     print(f'Menggunakan k: {k}, dengan rata-rata akurasi:{(sum(accs)/3)}%')

     Menggunakan k: 5, dengan rata-rata akurasi: 0.02%
```

Untuk melihat Output setiap baris code bisa melalui link Google Collab di bawah ini:

https://colab.research.google.com/drive/1SqPAII3V3GpYAHGpVolyDXs_A66tEKeP?usp=sharing

2.2 *Naïve Bayes*

A. Pengertian

Naive Bayes merupakan metode pengklasifikasian probabilistik sederhana. Metode ini akan menghitung sekumpulan probabilitas dengan menjumlahkan frekuensi dan kombinasi nilai dari dataset yang diberikan. Metode *Naive Bayes* menganggap semua atribut pada setiap kategori tidak memiliki ketergantungan satu sama lain (independen) (Nafalski & Wibawa, 2016).

Keuntungan penggunaan *Naive Bayes* yaitu hanya memerlukan sejumlah kecil data latih untuk menentukan parameter mean dan varians dari variabel yang diperlukan untuk klasifikasi (Palaniappan dan Awang, 2008). *Naive Bayes* merupakan metode supervised document classification yang berarti membutuhkan data training sebelum melakukan proses klasifikasi.

B. Cara Kerja

Pengklasifikasi Naive Bayes bekerja berdasarkan prinsip bahwa tidak adanya atau adanya fitur tertentu tidak bergantung pada ada atau tidaknya fitur lain yang lain yang diberi label kelas. Berbagai penelitian telah menunjukkan bahwa Pengklasifikasi Naive Bayes dapat bersaing dengan pengklasifikasi-pengklasifikasi lainnya
Pengklasifikasi Naive Bayes mengikuti model bersyarat yaitu:

$$P(C|F_1, \ldots . F_n)$$

Di mana C adalah variabel kelas dependen, masalahnya ketika ada data yang sangat besar atau ketika vektor fitur terlalu besar, penggunaan model bersyarat tidak dapat dilakukan,
Jadi kami memformulasikan model agar lebih efisien dengan dengan menggunakan teorema Bayes, yaitu:

$$P(C|F_1, \ldots . F_n) = \frac{p(C)p(F_1|C)p(F_2|C), \ldots . , p(F_n|C)}{Evidence}$$

$$Evidence = p(C_1)p(F_1|C_1)..p(F_n|C_1) + p(C_2)p(F_1|C2)..p(F_n|C_2) + p(C_n)\, p(F_1|C_n) \ldots p(F_n|C_n).$$

$$Posterior\ Probability = \frac{(Prior\ Probability\ x\ Class\ Conditional\ Probability)}{Evidence}$$

Dalam pelatihan pengklasifikasi ini, rata-rata dan varians darisetiap atribut dicari. Nilai-nilai ini kemudian digunakan dalam prosedur pengujian. Dalam prosedur pengujian posterior probabilitas dari sampel pengujian tersebut ditemukan terhadap setiap kelas. Sampel uji termasuk dalam kelas yang memiliki probabilitas posteriornya lebih besar.

C. Implementasi Code

```
#Mengimpor library
import pandas as pd
import requests
import numpy as np
import math
```

```
#membaca dataset dari URL ke dalam DataFrame
url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/arrhythmia/arrhythmia.data'
df = pd.read_csv(url, header=None)
```

```
df.head()
```

```
# Mengganti beberapa nama kolom dalam DataFrame dengan menggunakan dictionary
df = df.rename(columns={0: 'Age',1: 'Gender',2: 'Height',3: 'Weight',4:
'QRS_duration',5: 'P-R_interval',6: 'Q-T_interval', 7: 'T_interval', 8:
'P_interval',
```

```
                        9: 'QRS', 10: 'T', 11:'P', 12: 'QRST', 13: 'J', 14:
'Heart_rate'})

# Menampilkan nama-nama kolom dalam DataFrame
df.columns.values

# Menampilkan informasi tentang DataFrame, termasuk jumlah entri dan tipe
data kolom
df.info()

# Menghapus kolom 'QRS' dari DataFrame
df = df.drop(['QRS'], axis = 1)

# Membagi dataset menjadi beberapa lipatan (folds) untuk validasi silang
fold1 = (df.iloc[0:150].reset_index(drop=True),
df.iloc[300:452].reset_index(drop=True))
fold2 = (df.iloc[150:300].reset_index(drop=True), pd.concat([df.iloc[0:150],
df.iloc[300:452]]).reset_index(drop=True))
fold3 = (df.iloc[300:452].reset_index(drop=True),
df.iloc[0:150].reset_index(drop=True))

#Fungsi Normalisasi
def norm(allData, columnTarget):
  for column in columnTarget:
    allData[column] = (allData[column]-allData[column].min()) /
(allData[column].max()-allData[column].min())

  return allData

# Melakukan normalisasi pada kolom 'Age', 'Gender', dan 'QRS_duration' dalam
DataFrame
testStartID = df.index.stop
allData = pd.concat([df])
allData = norm(allData, ['Age','Gender','QRS_duration'])
normTrain= allData.iloc[:testStartID].drop('Height', axis=1)
normTrain

# Menampilkan beberapa baris pertama dari hasil normalisasi
normTrain.head()

#Fungsi Standarisasi
def Standarisasi(allData, columnTarget):
  for column in columnTarget:
    allData[column] = (allData[column] - allData[column].mean()) /
allData[column].std()
```

```
  return allData
```

```
# Melakukan normalisasi pada kolom 'Age', 'Gender', dan 'QRS_duration' dalam
DataFrame
testStartID = df.index.stop
allData = pd.concat([df])
allData = Standarisasi(allData, ['Age','Gender','QRS_duration'])
stdrTrain= allData.iloc[:testStartID].drop('Height', axis=1)
stdrTrain
```

```
# Menampilkan beberapa baris pertama dari hasil standarisasi
stdrTrain.head()
```

```
# Fungsi untuk menghitung probabilitas suatu nilai dengan menggunakan mean
dan standard deviation
def probality(mean, std, x):
  exponent = math.exp(-((x-mean)**2 / (2 * std**2)))
  return (1/(math.sqrt(2*math.pi)*std)) * exponent
```

```
# Fungsi untuk membagi DataFrame menjadi DataFrames terpisah berdasarkan
nilai pada kolom target
def splitTruth(df, columnTarget):
  truthData = []
  for truth in df[columnTarget].unique():
    truthData.append(df.where(df[columnTarget] == truth).dropna())
  return truthData

yesData, noData= splitTruth(df, columnTarget='Gender')
```

```
#Membaca nilai 1 yang disimpan di yesData
yesData.head()
```

```
#Membaca nilai 0 yang disimpan di noData
noData.head()
```

```
# Fungsi untuk menghitung mean dari kolom-kolom tertentu untuk nilai '1' dan
'0' secara terpisah
def findMean(yesData, noData, columnTarget):
  yesMean = dict()
  noMean = dict()
  for column in columnTarget :
    yesMean[column] = yesData[column].mean()
    noMean[column] = noData[column].mean()
  return yesMean, noMean
```

```
# Menghitung mean untuk kolom 'Age', 'Gender', dan 'QRS_duration' untuk nilai
'1' dan '0'
yesMean, noMean = findMean(yesData, noData, columnTarget =
['Age','Gender','QRS_duration'])

print(f"Mean Result\n1 : {yesMean}\n0 : {noMean}")
```

```
# Fungsi untuk menghitung standard deviation dari kolom-kolom tertentu untuk
nilai '1' dan '0' secara terpisah
def findStd(yesData, noData, columnTarget):
  yesStd = dict()
  noStd = dict()
  for column in columnTarget :
    yesStd[column] = yesData[column].std()
    noStd[column] = noData[column].std()

  return yesStd, noStd

yesStd, noStd = findStd(yesData, noData, columnTarget =
['Age','Gender','QRS_duration'])
print(f"Mean result\n1 : {yesStd}\n0 : {noStd}")
```

```
# Fungsi untuk melakukan prediksi nilai pada kolom target berdasarkan Naive
Bayes classifier menggunakan mean dan standard deviation yang telah dihitung
def prediction(yesMean, yesStd, noMean, noStd, target, columnTarget,
truthColumn):
    result = []
    for i in range(len(target)):
      yesResult = 1
      noResult = 1
      for column in columnTarget:
        yesResult *= probality(yesMean[column], yesStd[column],
target[column].iloc[i])
        noResult *= probality(noMean[column], noStd[column],
target[column].iloc[i])

      result.append({'QRS Duration' : target['QRS_duration'].iloc[i],
                     'Yes Probability' : "{}".format(yesResult),
                     'No Probability' : "{}".format(noResult),
                     'Prediction Result' : int(yesResult > noResult),
                     'Ground Truth' : target[truthColumn].iloc[i]})

    return result
```

```
result = []
target = df
result = prediction(yesMean, yesStd, noMean, noStd, target, columnTarget =
['QRS_duration'], truthColumn='Gender')

for p in result:
  print(p)
```

```
# Fungsi untuk membagi dataset menjadi training set dan validation set
berdasarkan persentase dan lokasi
def folding(dataset, trainingPercentage, location, shuffle:bool):
    lengthTraining = int(len(dataset)*trainingPercentage/100)
    # randomize the the data position
    if(shuffle):
        dataset = dataset.sample(frac=1).reset_index(drop=True)
    naive = []
    validation = []
    if(location == 'left'):
        naive, validation =
dataset.iloc[:lengthTraining].reset_index(drop=True),
dataset.iloc[lengthTraining:].reset_index(drop=True)
    elif(location == 'right'):
        validation,naive = dataset.iloc[:abs(lengthTraining-
len(dataset))].reset_index(drop=True), dataset.iloc[abs(lengthTraining-
len(dataset)):].reset_index(drop=True)
    elif(location == 'middle'):
        naive = dataset.iloc[int(abs(lengthTraining-
len(dataset))/2):len(dataset)-int(abs(lengthTraining-len(dataset))/2)]
        validation = pd.concat([dataset.iloc[:int(abs(lengthTraining-
len(dataset))/2)],dataset.iloc[len(dataset)-int(abs(lengthTraining-
len(dataset))/2):]])
    return naive,validation

dataTrain, validationData = folding(df.copy(), 70, 'left', shuffle=True)
```

```
# Menampilkan training set
dataTrain
```

```
# Menampilkan validation set
validationData
```

```
dataTrain, validationData = splitTruth(df, columnTarget='Gender')
```

```
# Split yes and no
yesDataTrain, noDataTrain = splitTruth(df, columnTarget='Gender')
# Find mean
```

```
yesMeanTrain, noMeanTrain = findMean(yesDataTrain, noDataTrain,
columnTarget=['Age','Weight','QRS_duration'])
# Find standard deviation
yesStdTrain, noStdTrain = findStd(yesDataTrain, noDataTrain,
['Age','Weight','QRS_duration'])

result = []
target = validationData # your ground truth data
# Melakukan prediksi pada validation set menggunakan Naive Bayes classifier
yang telah dilatih pada training set
result = prediction(yesMeanTrain, yesStdTrain, noMeanTrain, noStdTrain,
target, columnTarget=['Age','Weight','QRS_duration'], truthColumn='Gender')

for p in result:
    print(p)
```

```
# Fungsi untuk menghitung dan menampilkan confusion matrix, akurasi,
precision, recall dan F1 Score
def confussionMatrix(result):
    TP = 0
    FP = 0
    TN = 0
    FN = 0
    x = True
    for i in result:
        if(i['Ground Truth'] == '?'):
            x = False
            break
        elif((i['Prediction Result'] == 1) and (i['Prediction Result'] ==
i['Ground Truth'])):
            TP += 1
        elif((i['Prediction Result'] == 0) and (i['Prediction Result'] ==
i['Ground Truth'])):
            TN += 1
        elif((i['Prediction Result'] == 1) and (i['Prediction Result'] !=
i['Ground Truth'])):
            FP += 1
        elif((i['Prediction Result'] == 0) and (i['Prediction Result'] !=
i['Ground Truth'])):
            FN += 1
    if x:
        accuracy = (TP + TN) / (TP + TN + FP + FN)
        precision = TP / (TP + FP)
        recall = TP / (TP + FN)
        f1_score = 2 * (precision * recall) / (precision + recall)
```

```
        print(f"\nTP: {TP}  FN: {FP}\nTN: {TN}  FN: {FN}")
        print(f"Accuracy: {accuracy * 100}%")
        print(f"Precision: {precision * 100}%")
        print(f"Recall: {recall * 100}%")
        print(f"F1 Score: {f1_score * 100}%")
    else:
        print("\nCannot process the confusion matrix with unknown Ground
Truth!")

confussionMatrix(result)
```

Output Akhir:

```
TP: 62  FN: 0
TN: 0  FN: 187
Accuracy: 24.899598393574294%
Precision: 100.0%
Recall: 24.899598393574294%
F1 Score: 39.871382636655945%
```

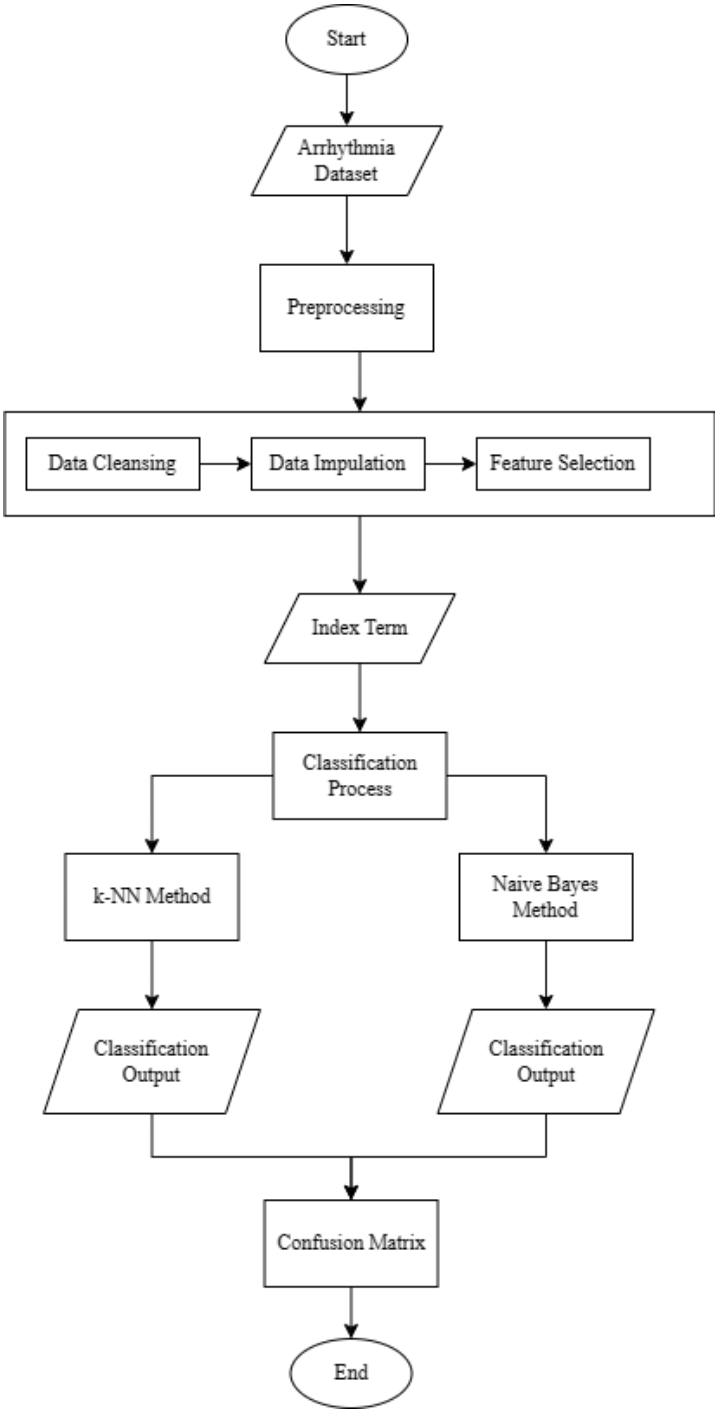
Untuk melihat Output setiap baris code bisa melalui link Google Collab di bawah ini:

https://colab.research.google.com/drive/1doajZJSw6z3bBuRpahDOvv5VcadK1DE1?usp=sharing

Link Github:

https://github.com/ditorifadli/TUBES_AI-10

2.3 Alur Implementasi *K-Nearest Neighbor* (KNN) dan *Naive Bayes*



BAB III

EVALUASI HASIL DAN DISKUSI

Berikut ini adalah hasil dan grafik yang diperoleh dalam simulasi kami lakukan:

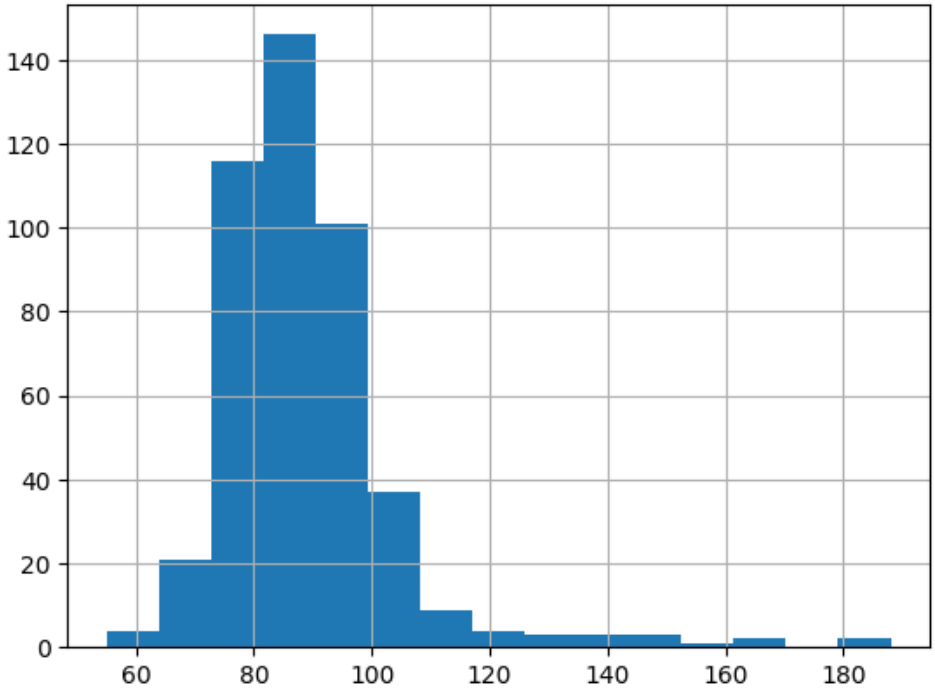
Algoritma	Accuracy
***K-Nearest Neighbor* (KNN)**	**0,2%**
Naïve Bayes	**24.899598393574294%**

3.1 Pengukuran Performansi

Performansi kedua algoritma ini diukur menggunakan Precision, Recall, F1, dan Accuration dengan rumus sebagai berikut:

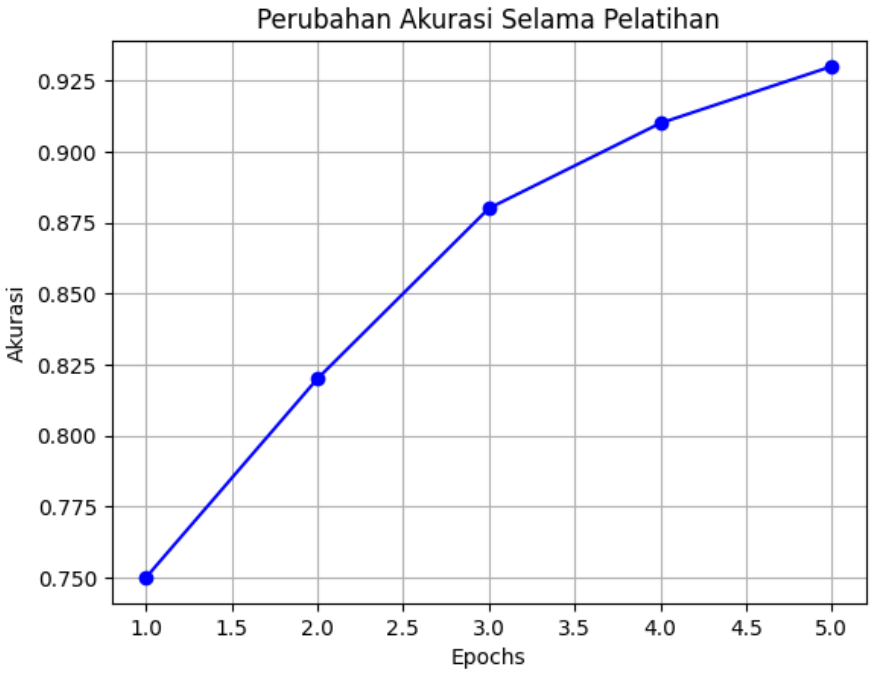
Accuracy = $\frac{TP + TN}{TP+FP+TN+FN}$

3.2 Hasil *K-Nearest Neighbor* (KNN)



Gambar 4. Grafik akurasi kNN

3.3 Hasil *Naïve Bayes*



Gambar 4. Grafik akurasi Naïve Bayes

BAB IV

KESIMPULAN DAN SARAN

4.1 Kesimpulan

EDA adalah pendekatan analisis data yang bertujuan untuk memahami karakteristik dasar dari data sebelum melakukan pemrosesan lebih lanjut. Tujuan utama EDA adalah mengungkap pola, tren, anomali, dan informasi penting lainnya yang terkandung dalam data. Melalui EDA, kita dapat menjelajahi struktur data dengan menggunakan teknik visualisasi seperti histogram, scatter plot, box plot, dan sebagainya. Hal ini membantu kita untuk memahami distribusi data, korelasi antar variabel, serta adanya outliers atau data yang tidak biasa.

EDA juga melibatkan analisis statistik deskriptif untuk menghitung ukuran-ukuran seperti mean, median, modus, dan deviasi standar. Ini memberikan informasi ringkas tentang data dan membantu dalam pemahaman awal tentang variabilitas dan kecenderungan dalam dataset.
EDA juga membantu dalam mengidentifikasi hubungan linier atau non-linier antara variabel dan menentukan apakah ada faktor yang mempengaruhi satu variabel terhadap variabel lainnya.

Dengan menggunakan algoritma kNN dan *Naive Bayes*, kami dapat menjalankan EDA sehingga mendapatkan apa yang perlu kita cari. Dari hasil yang ditunjukkan di atas, jelas bahwa kNearest Neighbor memberikan kami akurasi maksimum dalam mendeteksi aritmia. Sedangkan akurasi Naive Bayes lebih rendah dari metode kNN. Di masa depan, dapat dilakukan lebih banyak pekerjaan untuk meningkatkan klasifikasi aritmia, seperti membangun model hibrida yang mengklasifikasikan berdasarkan akurasi tertinggi dari klasifikasi yang berbeda.

4.2 Saran

Ketika melakukan EDA lebih baik menggunakan klasifikasi kNN karena paling efektif. Dengan melakukan EDA secara efektif, kita dapat mengurangi risiko kesalahan analisis, memperoleh wawasan yang lebih baik tentang data, dan mempersiapkan data dengan baik sebelum memasuki tahap pemodelan atau analisis lebih lanjut.

BAB V

DAFTAR PUSTAKA

[1] Samad, Saleha, et al. "Classification of arrhythmia." *International Journal of Electrical Energy* 2.1 (2014): 57-61.

[2] Wang, Lishan. "Research and implementation of machine learning classifier based on KNN." *IOP Conference Series: Materials Science and Engineering*. Vol. 677. No. 5. IOP publishing, 2019.

[3] Zhang, Min-Ling, and Zhi-Hua Zhou. *"A k-nearest neighbor based algorithm for multi-label classification."* 2005 IEEE international conference on granular computing. Vol. 2. IEEE, 2005.

LINK PRESENTASI

https://drive.google.com/drive/folders/1i6S8u-iKS2ZUH6QWdRDUS2TYJhLdYD2c?usp=sharing