PROPOSAL PROYEK DATA MINING
DIAGNOSIS SERANGAN JANTUNG DENGAN DATA MINING

Disusun oleh :

KELOMPOK 6

Nikolaus Filbert	C14200030
Samuel Christopher	C14200037
Rainer Trijuadi	C14200046
Alan Satria	C14200196

FAKULTAS TEKNOLOGI INDUSTRI

PROGRAM STUDI SISTEM INFORMASI BISNIS

UNIVERSITAS KRISTEN PETRA

BAB I
INTRODUCTION

1.1 Latar Belakang

Serangan jantung adalah gangguan jantung serius ketika otot jantung tidak mendapat aliran darah. Kondisi ini akan mengganggu fungsi jantung dalam mengalirkan darah ke seluruh tubuh. Serangan jantung dapat menyebabkan kematian bila tidak segera tertangani. Menurut data yang dikeluarkan oleh WHO pada tahun 2021, kematian akibat serangan jantung mencapai angka 17,8 juta kematian atau satu dari tiga kematian di dunia setiap tahun disebabkan oleh penyakit jantung. Padahal dengan adanya deteksi dini, dapat meningkatkan harapan hidup pasien. Jika dideteksi dini, langkah-langkah medis dapat dilakukan sebelum bertambah parah dan kondisi pasien memburuk. Namun sayangnya, cara ini jarang dipakai, terutama di rumah sakit dengan fasilitas kurang memadai yang mendiagnosis berlandaskan kepada keputusan dan pengalaman dokter. Industri kesehatan memiliki banyak data pasien yang berkaitan dengan penyakit jantung. Dengan laju perkembangan teknologi yang pesat, data mining dapat menjadi alternatif solusi untuk masalah deteksi dini penyakit jantung. Dengan data mining, data-data tersebut dapat diproses dan hasilnya digunakan untuk mendeteksi dini penyakit jantung. Dengan pengembangan lebih jauh dan implementasi nya yang lebih umum, akan mengurangi kasus penyakit jantung di masa mendatang.

1.2 Rumusan Masalah

- Apakah ada hubungan antara kondisi tubuh tertentu dengan terkenanya seseorang penyakit serangan jantung?
- Seberapa akurat kondisi tubuh yang mengindikasikan seseorang terkena penyakit jantung?

1.3 Tujuan

- Untuk mengetahui apakah ada hubungan antara kondisi tubuh tertentu dengan terkenanya penyakit serangan jantung.
- Mendapatkan tingkat akurasi dari faktor yang berkemungkinan besar memberikan

indikasi dari penyakit jantung.

- Memberikan visualisasi data pada user yang menginginkan total dari penyakit jantung.

1.4 Manfaat

- Orang dengan gejala seperti pada dataset dapat lebih mawas diri dan dapat menjaga kondisi tubuh mereka demi menghindari probabilitas terkena penyakit serangan jantung
- Tim medis dapat memberikan tindakan medis sesegera mungkin setelah dilakukan deteksi dini pada pasien dengan ciri-ciri seperti pada dataset

1.5 Ide Penyelesaian

Untuk meminimalisir dan meningkatkan kewaspadaan akan penyebab terjadinya penyakit jantung, kami membuat sebuah model klasifikasi yang ditujukan untuk melakukan deteksi dini pada orang-orang yang memiliki ciri-ciri tertentu yang berujung pada timbulnya penyakit jantung. Dengan adanya deteksi dini, diharapkan pasien dan juga tim medis dapat mengambil langkah medis yang tepat dan dapat teratasi dengan segera. Kelompok kami akan menggunakan ID3 Algorithm untuk pengimplementasian deteksi dini yang akan berkaitan erat dengan data-data pasien yang ada.

BAB II
LANDASAN TEORI

2.1 Metode

- **Bahasa Pemrograman**

  Bahasa pemrograman yang akan kami gunakan untuk implementasi deteksi dini pasien penyakit jantung ini adalah dengan menggunakan bahasa pemrograman Python dengan menggunakan Google Colaboratory.

- **Algoritma yang Digunakan**

  Kelompok kami akan menggunakan ID3 Algorithm untuk mengklasifikasikan dari dataset yang ada, apakah pasien terkait tergolong memiliki ciri-ciri yang menunjang timbulnya penyakit jantung atau tidak.

- **Cara Kerja Algoritma**

  ID3 Algorithm merupakan sebuah algoritma pembentukan *decision tree* untuk proses klasifikasi yang didasarkan pada perhitungan *entropy* dan *information gain* atribut pada dataset yang akan didasarkan pada *training data*. Untuk *root node* pada *decision tree* akan diambil dari perhitungan *information gain* yang paling tinggi. Kemudian untuk *node* selanjutnya juga tetap akan diambil dari perhitungan *information gain* yang paling tinggi yang akan juga didasarkan dari *root node* yang sudah didapatkan. ID3 Algorithm akan mendeskripsikan kriteria-kriteria orang yang terkena penyakit jantung melalui *decision tree* yang telah dimodelkan tersebut.

2.2 *Data Preprocessing*

- ***Data Cleaning***

  Dalam *data cleaning*, atribut data yang kosong akan diganti dengan nilai mean atau median untuk atribut tersebut. Hal ini juga akan ditinjau lagi dari dataset, jika dataset memiliki *outlier*, maka proses *data cleaning* akan menggunakan median. Jika suatu *row* memiliki banyak atribut yang kosong, maka akan dihapus. Untuk menangani noisy data, akan lebih efektif apabila menggunakan *Binning Method* untuk mengelompokkan data numerik di tiap atribut.

- ***Data Discretization***

  Membuat data menjadi data diskrit (data dikelompokkan sesuai dengan interval *value*, sebagai contoh, jika *value* kurang dari 80 maka masuk dalam kategori rendah, jika *value* sama dengan 80 atau kurang dari 120 maka masuk dalam kategori normal, dan sebagainya). *Data discretization* dilakukan dengan tujuan untuk memberikan konteks, jika hanya angka maka akan susah untuk menentukan arti dari angka tersebut, akan lebih mudah jika dilakukan pengkategorian tersebut yang sekaligus menjelaskan arti dari angka tersebut. Hal ini juga akan membantu untuk menghemat waktu ketika sesuatu dikategorikan.

BAB III
METODOLOGI PENELITIAN

3.1 Desain Sistem

Untuk meminimalisir dan meningkatkan kewaspadaan akan penyebab terjadinya penyakit jantung, kami membuat sebuah model klasifikasi yang ditujukan untuk melakukan deteksi dini pada orang-orang yang memiliki ciri-ciri tertentu yang berujung pada timbulnya penyakit jantung. Kelompok kami akan menggunakan ID3 Algorithm untuk pengimplementasian deteksi dini yang akan berkaitan erat dengan data-data pasien yang ada. Dataset yang kelompok kami gunakan bersumber dari Kaggle, yaitu *heart disease dataset* yang disebabkan oleh *cardiovascular disease*. Hasil penelitian yang kelompok kami lakukan nantinya adalah klasifikasi apakah pasien dengan ciri-ciri yang ada seperti pada dataset tergolong pasien yang memiliki ciri-ciri penyakit jantung atau tidak.

About Dataset

Cardiovascular diseases are the number 1 cause of death worldwide, claiming an estimated 17.9 million lives each year, which represents 31% of all deaths worldwide. Four out of every 5 deaths from cardiovascular disease are due to heart attacks and strokes, and a third of these deaths occur prematurely in people under 70 years of age.

People with cardiovascular disease or at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidemia, or established disease) need early detection and management, where a machine learning model can be of great benefit.

- Measures of 11 variables that characterize each sample (the features of the problem):
  - 1 - Age: patient's age (years)
  - 2 - Sex: patient's sex (M: Male, F: Female)
  - 3 - ChestPainType: chest pain type (TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic)
  - 4 - RestingBP: resting blood pressure (mm Hg)
  - 5 - Cholesterol: serum cholesterol (mm/dl)
  - 6 - FastingBS: fasting blood glucose (1: if FastingBS > 120 mg/dl, 0: otherwise)
  - 7 - RestingECG: Resting ECG results (Normal: normal, ST: with ST-T wave abnormality, LVH: showing probable or definite left ventricular hypertrophy by Estes criteria)
  - 8 - MaxHR: maximum heart rate reached (Numeric value between 60 and 202)
  - 9 - ExerciseAngina: exercise-induced angina (Y: Yes, N: No)
  - 10 - Oldpeak: old peak = ST (Numerical value measured in depression)
  - 11 - ST_Slope: the slope of the peak exercise ST segment (Up upsloping, Flat: flat, Down downsloping)

  In addition, there is the response variable, which in this case is a binary variable:
- 12 - HeartDisease: output class (1: heart disease, 0: normal)

3.2 Jadwal Kegiatan

12 November 2022	Mencari dataset
22 November 2022	Penyusunan proposal
	Proses pengerjaan *project* dalam *code*
1 Desember 2022	Melakukan *data cleaning* dan *data discretization*
8 Desember 2022	Penerapan proses *classification* dan implementasi ID3 Algorithm dalam bentuk *code*
10 Desember 2022	Pengisian hasil analisis data pada proposal
16 Desember 2022	Presentasi hasil proyek
18 Desember 2022	Penambahan *code* untuk *classification* pada *project*
22 Desember 2022	Finalisasi proposal *project*

3.3 Pembagian Tugas

- Nikolaus Filbert: Mencari dataset, *data cleaning* dan *data discretization*, implementasi ID3 Algorithm.
- Samuel Christopher: Mencari dataset, membuat proposal, pembuatan proses *classification* dalam *code*, menulis hasil analisa di proposal.
- Rainer Trijuadi: Mencari dataset, *data cleaning* dan *data discretization*, implementasi ID3 Algorithm.
- Alan Satria: Mencari dataset, membuat proposal, pembuatan proses *classification* dalam *code*, menulis hasil analisa di proposal.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Penjelasan

Kelompok kami mengimplementasikan 2 pendekatan *classification*. Pendekatan yang pertama adalah *classification* dengan menggunakan *manual code* dengan Python, sedangkan pendekatan yang kedua adalah dengan memanfaatkan *library* Python yang telah ada.

4.1.1 *Data Preprocessing*

4.1.1.1 *Data Cleaning*

Dataset akan dilakukan proses *data cleaning* terlebih dahulu. Proses *data cleaning* yang dilakukan pada *dataset* adalah sebagai berikut.

1. Hapus data yang sekiranya merupakan *noisy data*. Dalam *dataset*, terdapat 1 baris dari *dataset* yang merupakan *noisy data*. Karena merupakan sebuah *noisy data*, maka kami menghapus 1 baris data tersebut.
2. *Rename* pada kolom Cholesterol menjadi Triglycerides. Tidak ada perbedaan yang signifikan antara Cholesterol dan Triglycerides, Triglycerides merupakan salah satu jenis kolesterol yang ada, sehingga berdasarkan angka yang kami miliki pada *dataset*, kami memutuskan untuk menamai kembali kolom Cholesterol menjadi Triglycerides dengan maksud untuk mendetailkan jenis kolesterol apa yang dimaksudkan pada *dataset*.
3. *Handle missing data* yang ada pada kolom Triglycerides dengan menggunakan median. Ada beberapa baris yang kosong pada kolom Triglycerides, untuk mengatasi ini, maka data yang kosong akan diisi dengan menggunakan median dari kolom Triglycerides.
4. Modifikasi kolom Oldpeak. Indikator angka yang terdapat pada kolom Oldpeak akan dilakukan pembagian 10 dengan tujuan untuk mendapatkan rentang Oldpeak yang sesuai dengan tabel klasifikasi Oldpeak.

5. Modifikasi kolom Age. Kolom age akan diganti isinya menjadi rentang usia, sehingga kolom Age dari yang semula berisi umur yang spesifik dari tiap baris *dataset*, diubah menjadi rentang usia per barisnya.

   Contoh:

   Age lama: 45

   Age baru (dengan menggunakan rentang usia): 41-50

4.1.1.2 *Data Discretization*

Setelah dilakukan proses *data cleaning*, *dataset* akan dikelompokkan sesuai dengan kelompok-kelompok yang telah ditentukan. Pengelompokkan data yang dilakukan terhadap *dataset* adalah sebagai berikut.

Ada 6 kolom yang dilakukan pengelompokkan:

1. *Age*
   - Jika angka di antara **21-30,** maka akan dikategorikan dalam kelompok umur **21-30**
   - Jika angka di antara **31-40,** maka akan dikategorikan dalam kelompok umur **31-40**
   - Jika angka di antara **41-50,** maka akan dikategorikan dalam kelompok umur **41-50**
   - Jika angka di antara **51-60,** maka akan dikategorikan dalam kelompok umur **51-60**
   - Jika angka di antara **61-70,** maka akan dikategorikan dalam kelompok umur **61-70**
   - Jika angka di antara **71-80,** maka akan dikategorikan dalam kelompok umur **71-80**
2. *Resting* BP (*blood pressure*)
   - Jika angka **kurang dari 80 (< 80)**, maka dikategorikan sebagai ***low***
   - Jika angka **antara 80 hingga 120 (>= 80 and < 120)**, maka

dikategorikan sebagai **normal**

- Jika angka **lebih dari sama dengan 120 (>= 120)**, maka dikategorikan sebagai ***high***

3. *Fasting* BS (*blood glucose*)
   - Jika angka sama dengan 0, maka dikategorikan sebagai **normal**
   - Jika angka sama dengan 1, maka dikategorikan sebagai **diabetes**
4. *Triglycerides*
   - Jika angka **kurang dari sama dengan 150 (<= 150)**, maka dikategorikan sebagai **normal**
   - Jika angka **antara 150 hingga sama dengan 200 (> 150 and <= 200)**, maka dikategorikan sebagai ***borderline high*** (mencapai ambang batas tinggi)
   - Jika angka **antara 200 hingga sama dengan 500 (> 200 and <= 500)**, maka dikategorikan sebagai ***high***
   - Jika angka **lebih dari 500 (> 500)**, maka dikategorikan sebagai ***very high***
5. *Max* HR (*heart rate*)
   - Jika umur di antara **21-30:**
     - Jika angka **kurang dari 170 (< 170)**, maka dikategorikan sebagai ***low***
     - Jika angka **antara 170 hingga 200 (>= 170 and < 200)**, maka dikategorikan sebagai **normal**
     - Jika angka **lebih dari sama dengan 200 (>= 200)**, maka dikategorikan sebagai ***abnormal***
   - Jika umur di antara **31-40:**
     - Jika angka **kurang dari 162 (< 162)**, maka dikategorikan sebagai ***low***
     - Jika angka **antara 162 hingga 190 (>= 162 and < 190)**, maka dikategorikan sebagai **normal**
     - Jika angka **lebih dari sama dengan 190 (>= 190)**, maka

dikategorikan sebagai ***abnormal***

- Jika umur di antara **41-50:**
  - Jika angka **kurang dari 153 (< 153)**, maka dikategorikan sebagai ***low***
  - Jika angka **antara 153 hingga 180 (>= 153 and < 180)**, maka dikategorikan sebagai **normal**
  - Jika angka **lebih dari sama dengan 180 (>= 180)**, maka dikategorikan sebagai ***abnormal***
- Jika umur di antara **51-60:**
  - Jika angka **kurang dari 145 (< 145)**, maka dikategorikan sebagai ***low***
  - Jika angka **antara 145 hingga 170 (>= 145 and < 170)**, maka dikategorikan sebagai **normal**
  - Jika angka **lebih dari sama dengan 170 (>= 170)**, maka dikategorikan sebagai ***abnormal***
- Jika umur di antara **61-70:**
  - Jika angka **kurang dari 136 (< 136)**, maka dikategorikan sebagai ***low***
  - Jika angka **antara 136 hingga 160 (>= 136 and < 160)**, maka dikategorikan sebagai **normal**
  - Jika angka **lebih dari sama dengan 160 (>= 160)**, maka dikategorikan sebagai ***abnormal***
- Jika umur di antara **71-80:**
  - Jika angka **kurang dari 128 (< 128)**, maka dikategorikan sebagai ***low***
  - Jika angka **antara 128 hingga 150 (>= 128 and < 150)**, maka dikategorikan sebagai **normal**
  - Jika angka **lebih dari sama dengan 150 (>= 150)**, maka dikategorikan sebagai ***abnormal***

6. *Oldpeak*
   - Jika angka **kurang dari 1,75 (< 1,75)**, maka dikategorikan

sebagai ***low***

- Jika angka **antara 1,75 hingga 3,75 (>= 1,75 and < 3,75)**, maka dikategorikan sebagai **normal**
- Jika angka **lebih dari sama dengan 3,75 (>= 3,75)**, maka dikategorikan sebagai ***terrible***

Hasil *dataset*: *manual code train dataset*

Proses pengelompokkan yang telah direalisasikan di atas **hanya berlaku untuk proses *classification* pendekatan pertama (*manual code*)**, untuk *classification* pendekatan kedua (menggunakan *library*) *dataset* harus diolah lebih lanjut dengan mengganti hasil *discretization* yang telah dilakukan pada semua kolom menjadi angka. **Berikut adalah hasil pengolahan *dataset* untuk *classification* pendekatan kedua.**

1. *Age Cluster*
   - Jika angka adalah **21-30,** maka akan diganti dengan angka 1
   - Jika angka adalah **31-40,** maka akan diganti dengan angka 2
   - Jika angka adalah **41-50,** maka akan diganti dengan angka 3
   - Jika angka adalah **51-60,** maka akan diganti dengan angka 4
   - Jika angka adalah **61-70,** maka akan diganti dengan angka 5
   - Jika angka adalah **71-80,** maka akan diganti dengan angka 6
2. *Chest Pain Type*
   - Jika tipe adalah **TA,** maka akan diganti dengan angka 1
   - Jika tipe adalah **ATA,** maka akan diganti dengan angka 2
   - Jika tipe adalah **NAP,** maka akan diganti dengan angka 3
   - Jika tipe adalah **ASY,** maka akan diganti dengan angka 4
3. *Resting* BP
   - Jika *resting* BP adalah **normal,** maka akan diganti dengan angka 1
   - Jika *Resting* BP adalah ***high*,** maka akan diganti dengan angka 2

4. *Triglycerides*
    - Jika *triglycerides* adalah **normal,** maka akan diganti dengan angka 1
    - Jika *triglycerides* adalah ***high*,** maka akan diganti dengan angka 2
    - Jika *triglycerides* adalah ***borderline high*,** maka akan diganti dengan angka 3
    - Jika *triglycerides* adalah ***very high*,** maka akan diganti dengan angka 4
5. *Fasting* BS
    - Jika *fasting* BS adalah **normal,** maka akan diganti dengan angka 1
    - Jika *triglycerides* adalah **diabetes,** maka akan diganti dengan angka 2
6. *Resting* ECG
    - Jika *resting* ECG adalah **normal,** maka akan diganti dengan angka 1
    - Jika *resting* ECG adalah **ST,** maka akan diganti dengan angka 2
    - Jika *resting* ECG adalah **LVH,** maka akan diganti dengan angka 3
7. *Max* HR
    - Jika *max* HR adalah ***low*,** maka akan diganti dengan angka 1
    - Jika *max* HR adalah **normal,** maka akan diganti dengan angka 2
    - Jika *max* HR adalah ***abnormal*,** maka akan diganti dengan angka 3
8. *Exercise Angina*
    - Jika *exercise angina* adalah **Y,** maka akan diganti dengan angka 1
    - Jika *exercise angina* adalah **N,** maka akan diganti dengan angka 2

9. *Oldpeak*
   - Jika *oldpeak* adalah ***low*,** maka akan diganti dengan angka 1
   - Jika *oldpeak* adalah **normal,** maka akan diganti dengan angka 2
   - Jika *oldpeak* adalah ***terrible*,** maka akan diganti dengan angka 3
10. ST *Slope*
   - Jika ST *slope* adalah ***up*,** maka akan diganti dengan angka 1
   - Jika ST *slope* adalah ***flat*,** maka akan diganti dengan angka 2
   - Jika ST *slope* adalah ***down*,** maka akan diganti dengan angka 3

Hasil *dataset*: Python *library train dataset*

4.1.2 *Classification Process* - Python *Code*

4.1.2.1 *Manual Code*

Untuk membuat *decision tree* dari *training dataset* 1 tersebut, kami menggunakan bantuan *library pandas* untuk *load dataset* ke Python *code*. *Dataset* yang akan dimasukkan akan diubah modelnya menjadi *dataframe*. *Dataset* dalam bentuk *dataframe* inilah yang nantinya akan diproses lebih lanjut.

```
import pandas as pd
df=pd.read_excel('train_data_2.xlsx')
```

Setelah *load dataset* yang akan digunakan, kami melakukan *drop* kolom *data* untuk menghapus kolom yang tidak digunakan. Pada kasus kali ini kami melakukan proses *drop* untuk kolom "Index", sehingga nantinya kolom "Index" akan dihapus dari *dataset* yang akan digunakan. 'axis =1' bermakna yang akan di-*drop* adalah sebuah kolom dan 'inplace = true" memiliki arti bahwa *dataset* nantinya tidak akan mengembalikan suatu *value*, namun akan langsung melakukan *update* data dalam *dataset*.

```
df.drop(['Index'], axis=1, inplace=True)
df
```

Setelah *dataset* berhasil untuk dimasukkan ke Python, kami membuat sebuah fungsi untuk menghitung entropy total dari *training dataset*. Variabel **heartDisease** merupakan variabel yang akan menyimpan *label* hasil akhir dari kolom "HeartDisease" pada *dataset* yang berisi nilai yang artinya menderita penyakit jantung (1) atau tidak (0). Variable **values** digunakan untuk menyimpan nilai unik dari variabel heartDisease dengan tipe *binary* 1 atau 0. *Binary* 1 berarti terkena penyakit jantung dan *binary* 0 berarti tidak terkena penyakit jantung. Dalam *function* akan dilakukan iterasi untuk mencari berapa banyak jumlah *row* data yang memiliki nilai ***heart disease*** 1 dan 0 dengan menggunakan .value_counts()[i] yang kemudian dibagi dengan total *data* untuk mendapatkan probabilitas yang nantinya akan dipakai untuk mencari entropy dengan rumus:

$$\text{Entropy (S)} = \sum_{i=1}^{n} - pi * \log_2 pi$$

```
# ===== MENGHITUNG ENTROPY TOTAL ====
def calculateEntropy(dataset):
  heartDisease = dataset.keys()[-1]
  values = dataset[heartDisease].unique()
  entropy = 0
  for i in values:
      probability=dataset[heartDisease].value_counts()[i]/len(da
      taset[heartDisease])
      entropy += -probability * np.log2(probability)

  return np.float(entropy)
```

Kemudian, juga terdapat *function* untuk menghitung entropy dari masing-masing atribut dari *dataset*. *Function* memiliki 2 parameter, parameter pertama merupakan tabel *dataset training* dan parameter kedua

merupakan atribut yang akan dihitung entropy-nya. Variabel **heartDesease** berfungsi untuk menyimpan data pada kolom terakhir, yaitu kolom "Heart Disease". Variabel **parentValues** untuk mencari nilai unik variabel heartDisease, yaitu 0 dan 1. Variabel **attributeValues** digunakan untuk mendapatkan semua *unique value* dari atribut yang diminta. Untuk masing-masing *value* atribut, akan dihitung berapa banyak *row* data yang memiliki atribut tersebut dan memiliki nilai parentValue sama dengan *counternya*, yaitu **i**. Langkah selanjutnya akan dicari entropy dengan rumus entropy dan ditambahkan 0,0001 untuk menghindari penyebut bernilai 0 dan berpotensi menimbulkan *zero division error* pada *code*. Setelah nilai atribut tersebut telah dicari semua berdasarkan nilai *parent*, maka akan dihitung entropy rata-rata dan akan disimpan pada variabel **avgEntropy**. Jika semua entropy untuk masing-masing atributnya telah dihitung, maka hasil *average* entropy untuk atribut tersebut telah didapatkan.

```
# ===== MENGHITUNG ENTROPY PER ATRIBUT =====
def calculateEntropyAttribute(dataset, attribute):
  heartDisease = dataset.keys()[-1]
  parentValues = dataset[heartDisease].unique()
  attributeValues = dataset[attribute].unique()
  avgEntropy = 0
  for i in attributeValues:
    entropy = 0
    for j in parentValues:
       a    =    len(dataset[attribute][dataset[attribute]    ==
i][dataset[heartDisease] == j])
       b = len(dataset[attribute][dataset[attribute] == i])
      probability = a/b
           entropy  +=  -probability  *  np.log2(probability  +
0.000001)
    avgEntropy += (b/len(dataset)) * entropy

  return np.float(avgEntropy)
```

Fungsi selanjutnya yang dibuat adalah fungsi untuk menghitung

jumlah *gain* dari setiap kolom yang ada dalam *dataset*. Variable **infoGain** merupakan sebuah *array* yang bertujuan untuk menyimpan penambahan *value* data sesuai dengan jumlah barisnya. Pada *function* juga dilakukan iterasi hingga sampai pada sebelum *dataset* kolom terakhir (dataset.keys()[:-1]). Dalam iterasi ini akan dimasukkan nilai *information gain* setiap atribut dengan melakukan pengurangan dari nilai entropy total dengan entropy masing-masing atribut. Hasil perolehan *information gain* yang akan di-*return* adalah perhitungan dari kolom paling awal hingga kolom kedua terakhir dari *dataset*, lalu dari semua perhitungan nilai *gain* akan di-*return* nilai *information gain* tertinggi dengan menggunakan syntax np.argmax.

```
# ===== MENGHITUNG INFORMATION GAIN TERTINGGI =====
def calculateInformationGain(dataset):
  infoGain = []
  for i in dataset.keys()[:-1]:
    infoGain.append(calculateEntropy(dataset) -
calculateEntropyAttribute(dataset, i))
  return dataset.keys()[:-1][np.argmax(infoGain)]
```

Function getValue Table berfungsi untuk melakukan filter dengan cara *drop* sebuah hasil yang hasil itu sendiri merupakan hasil *duplicate* yang sama sebelum membentuk sebuah *decision tree*.

```
def getValueTable(dataset, attribute, value):
   return dataset[dataset[attribute] == value].reset_index(drop
= True)
```

Fungsi berikutnya merupakan fungsi untuk membuat *decision tree* berdasarkan *dataset* dan menggunakan *functions* yang sebelumnya telah dibuat *Decision tree* akan disimpan dalam bentuk *dictionary*. Variabel **node** berfungsi untuk menyimpan hasil perhitungan *information gain* dengan menggunakan *function* calculateInformationGain. Variabel **attvalue** digunakan untuk menemukan elemen unik dari sebuah *array*. Kami mengurangi pengambilan data hingga sampai pada sebelum dataset

terakhir (dataset.keys()[:-1]). Dalam pengurangan ini, *code* akan membuat variable **tree** serta **tree[node]** yang berfungsi untuk menyimpan hasil akhir dari rekursi iterasi yang dilakukan. Kemudian kami melakukan iterasi (for i in attvalue), yang berarti selama i (*counter*) masih lebih kecil dari attvalue, maka iterasi ini akan tetap berlanjut untuk mendapatkan nilai yang paling besar. Dalam iterasi, variabel **subtable** berguna untuk menghapus data yang sama pada *value* data yang ada. Variabel **clvalue** dan **counts** merupakan variabel yang berfungsi untuk menentukan apakah data yang ada saat ini masuk dalam ketentuan yang berlangsung atau tidak, dengan tujuan akhir untuk membuat dan memvisualisasikan *decision tree*.

```
# ===== MEMBUAT TREE =====
def buildTree(dataset, tree = None):
  node = calculateInformationGain(dataset)
  attvalue = np.unique(dataset[node])
  heartDisease = dataset.keys()[-1]
  if tree is None:
    tree = {}
    tree[node] = {}
  for i in attvalue:
    subtable = getValueTable(dataset,node,i)
        Clvalue,  counts  =  np.unique(subtable[heartDisease],
return_counts = True)
    if len(counts) == 1:
      tree[node][i] = Clvalue[0]
    else:
      tree[node][i] = buildTree(subtable)
  return tree
```

Kami *declare* sebuah variabel **tree** yang akan memanggil fungsi buildTree dengan mengisi parameter *function* dengan *dataframe* (*training dataset*).

```
# ==== PRINT TREE ====
tree = buildTree(df)
```

Lalu, untuk menampilkan visualisasi *tree* yang telah dibuat di dalam fungsi buildTree dilakukan dengan cara menggunakan pprint *library*.

```
# ==== PRINT TREE ====
pprint.pprint(tree)
```

Visualisasi *tree* menggunakan *library* Pydot dan 2 *function*. *Function* pertama adalah draw(), dengan parameter parent_name dan child_name. *Function* kedua adalah visit(), dengan parameter node dan parent. *Function* draw() digunakan untuk membuat hubungan antara parent_node dan child_node. *Function* visit() digunakan untuk iterasi *dictionary* untuk melihat parent_node dan child_node yang ada. Jika value dari *key dictionary* merupakan sebuah *dictionary*, maka akan akan program akan membuat cabang *tree*, kemudian visit lagi ke cabang tersebut untuk dicek lagi *value*-nya hingga tidak ada lagi *value* dari *dictionary* yang berupa *dictionary*. Jika bukan merupakan dictionary, maka *node* tersebut akan di-*print* nama *node* sesuai dengan atribut dan juga bilangan 0 atau 1 sebagai penanda terkena penyakit jantung atau tidak.

```
# === Visualize Tree ===

def draw(parent_name, child_name):
    edge = pydot.Edge(parent_name, child_name
    graph.add_edge(edge

def visit(node, parent=None):
    for k,v in node.items():
        if isinstance(v, dict):
            if parent:
                draw(parent, k)
            visit(v, k)
        else:
            draw(parent, k)
            draw(str(k), k+'_'+str(v))
```

```
graph = pydot.Dot(graph_type='graph')
visit(tree)
graph.write_png('example4_graph.png')
```

Selanjutnya, untuk memprediksi *sample data,* kami membuat sebuah fungsi bernama predictTestCase yang memiliki parameter exampleData dan tree. Kami membuat sebuah variabel bernama **value** yang menampung *value* dari setiap data dalam *testing dataset* yang digunakan. Ada juga variabel **tree** yang akan menampung nilai dari parameter tree yang mengandung *node* dan *value*. Variabel **prediction** berfungsi untuk memberikan hasil prediksi dan *default value* akan diisi dengan angka 0. Kemudian, akan dilakukan pengecekan apakah tipe dari *tree* tersebut merupakan *dictionary* atau bukan, apabila benar maka variabel prediction akan diisi dengan fungsi predictTestCase, sedangkan bila tidak maka variabel prediction akan memiliki makna *tree*.

```
# ===== MEMPREDIKSI SAMPLE DATA =====
def predictTestCase(exampleData, tree):
  for node in tree.keys():
    value = exampleData[node]
    tree = tree[node][value]
    prediction = 0
    if type(tree) is dict:
      prediction = predictTestCase(exampleData, tree)
    else:
      prediction = tree

  return prediction
```

Berikut adalah penjelasan mengenai langkah-langkah untuk memprediksi *testing dataset*. Langkah pertama, kami melakukan *import file* Excel yang berisikan kolom kriteria pasien yang bersangkutan. *Dataset* akan diubah modelnya menjadi *dataframe* yang nantinya akan diproses dan diprediksi lebih lanjut. Kemudian, kami *load testing dataset*

dalam *code*. Variabel **results** merupakan sebuah *array* untuk menampung hasil prediksi dari *testing dataset*. Dilakukan iterasi sebanyak jumlah dari *testing dataset*. Selama iterasi akan dilakukan pengecekan dengan membuat variable **exampleData** yang akan berisikan data dengan lokasi yang telah ditentukan pada .iloc[i,:]. Lalu, dideklarasikan juga variable **prediction** yang akan memanggil fungsi predictTestCase dengan parameter exampleData dan tree. Hasil dari prediction tersebut akan dimasukan ke dalam array results. Setelah iterasi selesai, maka isi *array* results sudah terisi, untuk melihat hasilnya yaitu dengan *print* isi *array.*

```
df_test = pd.read_excel("test data paling baru ygy
8989.xlsx")
df_test

results = []
for i in range (len(df_test)):
  exampleData = df_test.iloc[i,:]
  prediction = predictTestCase(exampleData, tree)
  results.append(prediction)

print(results)
```

4.1.2.2 Python *Library*

```
from sklearn import tree
```

Dataset yang digunakan dalam pendekatan ini adalah *dataset* yang telah diolah lebih lanjut dengan mengubah masing-masing String yang ada pada masing-masing kolom menjadi angka.

```
#load data train
df_train_lib = pd.read_excel('3_train data buat
library.xlsx')
df_train_lib
```

Code pada gambar di bawah bertujuan untuk menyimpan label pada kolom HeartDisease (0 dan 1) yang mengkategorikan apakah pasien memiliki ciri yang menjerumus pada timbulnya penyakit jantung atau tidak. Label 0 dan 1 pada kolom HeartDisease yang sebelumnya merupakan sebuah kolom dari *dataframe*, diubah menjadi sebuah *array*.

```
# ===== MENGUBAH KOLOM HEART DISEASE MENJADI ARRAY =====
label = df_train_lib['HeartDisease']
label = label.to_numpy()
label
```

Berikutnya, seluruh *dataframe* akan diubah formatnya menjadi sebuah *array*.

```
# ===== MENGUBAH TRAIN DATAFRAME MENJADI ARRAY =====
df_train_lib = df_train_lib.to_numpy()
df_train_lib
```

```
clf_model = tree.DecisionTreeClassifier()
```

Setelah diubah format dari *dataframe* menjadi *array*, maka *training dataset* akan di-*train* dengan menggunakan clf_model.fit(df_train_lib, label). Parameter pertama adalah dengan memasukan keseluruhan *dataset* yang telah diubah format menjadi *array* dan parameter kedua adalah label 0 dan 1 yang telah disendirikan dalam satu variabel *array*.

```
# ===== TRAINING DATA =====
clf_model = clf_model.fit(df_train_lib, label)
```

```
# ===== VISUALISASI TREE =====
tree.plot_tree(clf_model)
```

Kami *load test dataset* yang telah sebagai bahan prediksi.

```
#load data test
df_test_lib = pd.read_excel('4_test data buat
library.xlsx')
df_test_lib
```

Test dataset kemudian diubah kembali formatnya menjadi sebuah *array* untuk dapat diprediksi.

```
# ===== MENGUBAH TEST DATAFRAME MENJADI ARRAY =====
df_test_lib = df_test_lib.to_numpy()
df_test_lib
```

Code di bawah merupakan *code* untuk memprediksi *test dataset* apakah pasien terkena penyakit jantung atau tidak.

```
# ===== MEMPREDIKSI TEST DATA =====
clf_model.predict(df_test_lib)
```

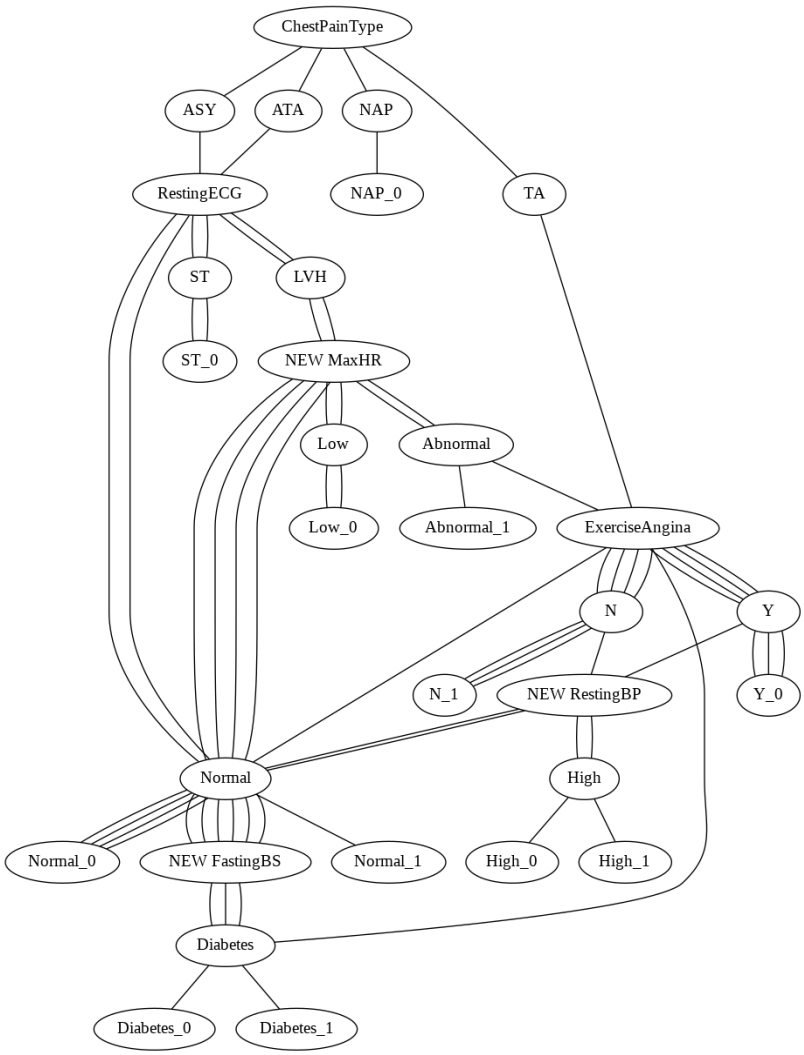
4.2 Hasil Percobaan

4.2.1 *Manual Code*

Berikut adalah hasil percobaan dengan menggunakan *testing dataset* sebanyak 17 *row data*.

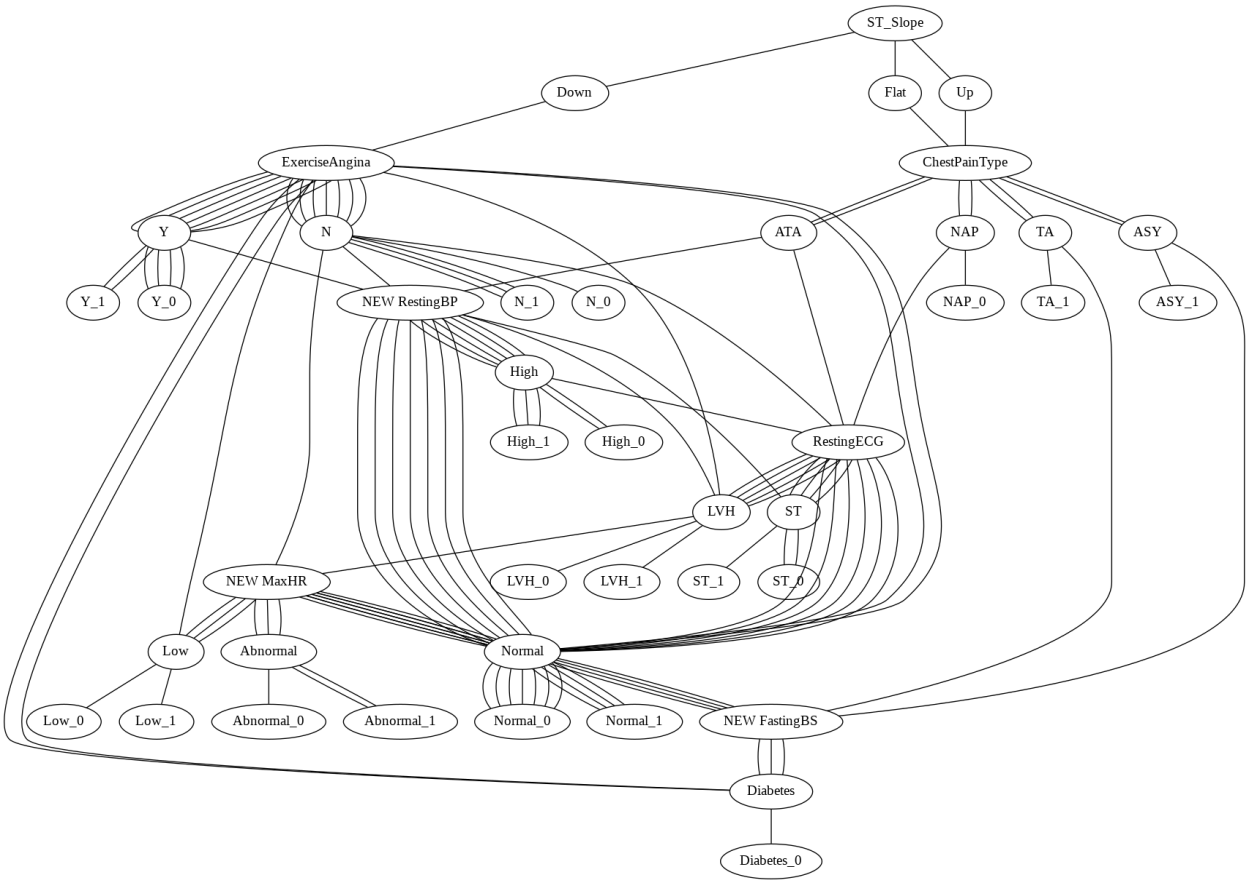
- **First 50 Row of Training Dataset**

  Hasil *accuracy test:* 64%



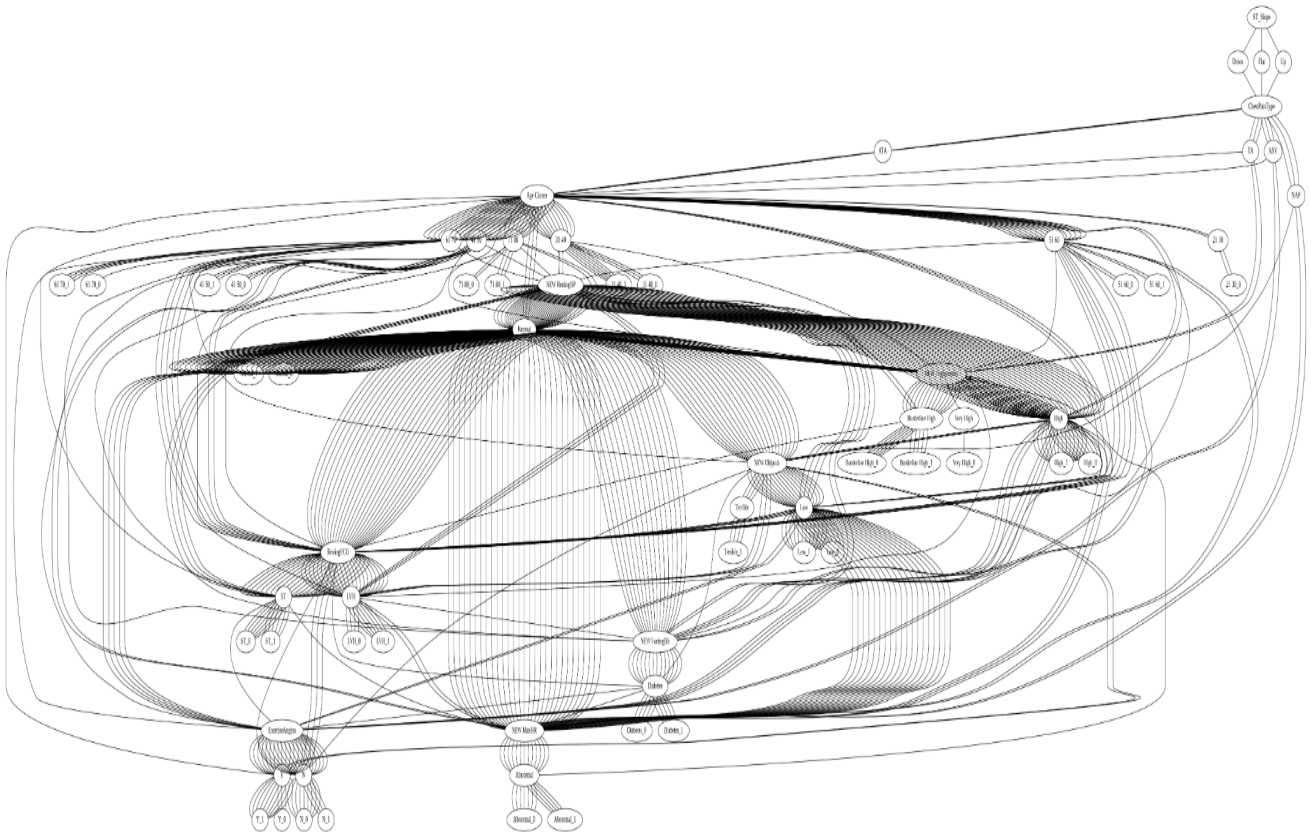
- **First 100 Row of Training Dataset**

  Hasil *accuracy test:* 88%

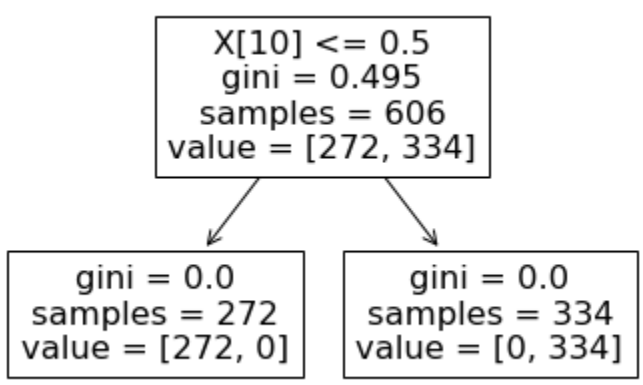


- **600 Row of Training Dataset**

  Hasil *accuracy test:* 88%



4.2.2 Python *Library*



BAB V
KESIMPULAN

5.1 Kesimpulan

Penyakit jantung akan menjadi semakin sulit diatasi jika terlambat dideteksi. Deteksi dini penyakit jantung adalah hal yang wajib dilakukan untuk memperbesar peluang untuk mengobati penyakit jantung. Namun, banyak kasus dimana hanya sedikit tenaga dan alat kesehatan di tempat dan jumlah pasien yang banyak sehingga akan membutuhkan waktu yang lama untuk mendiagnosis pasien. Dengan AI yang menggunakan ID3 decision tree dan di training dengan ribuan atau puluhan ribu data, maka kita bisa membuat program untuk melakukan quickscan terhadap informasi kesehatan pasien dan dengan cepat menentukan apakah mereka terkena penyakit jantung atau tidak.

Dari hasil penelitian yang menggunakan manual code program tanpa menggunakan library IPYNB dan menggunakan sebanyak 50, 100, dan 600 training data, serta 17 test data untuk pengujian. Dapat disimpulkan bahwa semakin banyak training data yang diberikan, maka semakin tinggi pula akurasi untuk menentukan seseorang terkena penyakit jantung atau tidak. Dalam uji coba yang kita lakukan kita mendapatkan sekitar 89% akurasi kebenaran. Jika dilihat dari dataset dan tree yang telah dibuat, marker utamanya adalah dari ST Slope, Chest Pain Type, dan kadar Trigliceride dimana ST Slope Flat, Chest Pain Type ASY, dan level Trigliserida tinggi berkemungkinan paling besar bahwa seseorang menderita penyakit jantung.

Sedangkan hasil penelitian yang menggunakan library IPYNB yang menggunakan sebanyak 600 training data serta 17 test data yang sama untuk pengujian mendapatkan hasil yang berbeda dengan manual code program yang telah dibuat. Hasil akhir menunjukan terdapat 2 hasil yang berbeda. Setelah melalui proses pengecekan didapatkan akurasi sebesar 100%. Dari sini dapat disimpulkan bahwa akurasi milik library python lebih besar daripada hasil manual code yang telah dibuat oleh tim kami, hal ini dapat terjadi dikarenakan adanya perbedaan algoritma code program yang dapat mempengaruhi hasil akhir.

Pada akhirnya, Decision Tree Algorithm yang di training menggunakan dataset

informasi kesehatan pasien bisa menjadi alat bantu bagi tenaga kesehatan untuk *Quickscan* atau pengecekan cepat yang melibatkan banyak orang dalam suatu waktu jika kekurangan kesehatan. Hasil dari AI ini tidak bisa menjadi patokan utama untuk menentukan seseorang terkena penyakit jantung atau tidak, karena pasti ada banyak faktor-faktor lain yang belum disebutkan karena keterbatasan akses informasi dan dataset. Dibutuhkan informasi dataset informasi kesehatan pasien dan atribut/faktor yang lebih banyak untuk membuat Decision Tree ini menjadi lebih akurat. Walau tidak bisa menjadi patokan utama, namun dari hasilnya itu bisa dijadikan patokan untuk melanjutkan pemeriksaan yang lebih lanjut. Masih banyak faktor lain yang dapat mempengaruhi hasil tree, yang kami namakan Faktor X. Factor X ini biasa seperti pengambilan sampel tidak sesuai kebutuhan (terlalu sedikit/banyak), human error, kesalahan dari alat ukur, dsb. Oleh karena itu, perlu waktu dan lebih banyak data yang lebih akurat untuk membuat tree ini menjadi lebih akurat.

BAB VI
REFERENSI

https://www.youtube.com/watch?v=YG_nOa6-6Q8&ab_channel=CSbySahilSharma

https://www.klikdokter.com/info-sehat/jantung/awas-kolesterol-tinggi-bisa-bikin-umur-lebih-pendek

https://www.kaggle.com/datasets/ronanazarias/heart-desease-dataset

https://www.cdc.gov/bloodpressure/about.htm#:~:text=%2F80%20mmHg.%E2%80%9D-,What%20are%20normal%20blood%20pressure%20numbers%3F,less%20than%20120%2F80%20mmHg.&text=No%20matter%20your%20age%2C%20you,pressure%20in%20a%20healthy%20range.

https://www.kaggle.com/questions-and-answers/205588

https://www.ijert.org/a-study-on-heart-disease-prediction-using-different-classification-models-based-on-cross-validation-method

https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/lipid-panel#:~:text=Here%20are%20the%20ranges%20for,or%20above%20240%20mg%2FdL

https://glints.com/id/lowongan/decision-tree-adalah/#.Y5iLV3b7RhE

https://towardsdatascience.com/id3-decision-tree-classifier-from-scratch-in-python-b38ef145fd9

https://www.datacamp.com/tutorial/decision-tree-classification-python

https://scikit-learn.org/stable/modules/tree.html

https://www.datacamp.com/tutorial/decision-tree-classification-python

https://www.forbes.com/health/healthy-aging/n

https://ijiset.com/vol2/v2s9/IJISET_V2_I9_54.pdf

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2376653

https://www.academia.edu/38012068/Laporan_Final_Project_Data_Mining_A_Charles_Rudiyanto_06211540000030

https://www.kaggle.com/datasets/ayessa/salary-prediction-classification