CASE BASED - 01

Mata Kuliah Pembelajaran Mesin

KODE DOSEN: GKL

Disusun oleh:

Berlian Muhammad Galin Al Awienoor (1301204378)

KELAS IF-44-10

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
2022/2022

PENDAHULUAN

A. Latar Belakang

Pada era industri 4.0, kemampuan mengolah data dan membuat model sudah seperti kewajiban. Kemajuan teknologi membuat pekerjaan pemrosesan data lebih mudah tetapi dengan kemajuan ini juga muncul banyak istilah baru. Pada perkembangan *Big Data* yang pesat seperti sekarang, kita sudah sering mendengar istilah *Machine Learning* ataupun *Artificial Intelligence*. Secara umum, model *Machine Learning* dapat dibedakan tergantung dari penggunaannya, seperti *Supervised* dan *Unsupervised,* yang merupakan istilah untuk memisahkan model dalam fungsi tertentu. Secara singkat, bisa dikatakan bahwa *Supervised Learning* adalah *machine learning model* yang membutuhkan data target sedangkan *Unsupervised Learning* tidak memerlukan data target.

Supervised Learning merupakan sebuah pemodelan dimana algoritmanya dapat membangkitkan suatu fungsi yang memetakan *input* ke *output* yang diinginkan. Pada *Supervised Learning* kita mengolah data yang memiliki label sehingga tujuan pengolahan tersebut adalah mengelompokkan data ke data yang sudah ada. Proses pengolahan data yang kita lakukan jika menggunakan *Supervised Learning* juga memerlukan data *training,* data *training* sendiri digunakan dalam memprediksi maupun mengklasifikasi data.

B. Rumusan Masalah

Pada tugas Case-Based kali ini diberikan file arrhythmia.data dengan kriteria seperti dibawah ini:

Data Set Characteristics:	Multivariate	**Number of Instances:**	452	**Area:**	Life
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	279	**Date Donated**	1998-01-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	410432

Dengan data tersebut, kita ditugaskan untuk menyelidiki masalah kualitas data yang telah diberikan, lalu menjelaskan keputusan mengenai pendekatan pra-pemrosesan data. Menjelajahi kumpulan data dengan meringkas data menggunakan statistik dan mengidentifikasi masalah kualitas data yang relevan.

C. Tujuan

Tujuan dari disusunnya laporan ini adalah sebagai berikut:

1. Untuk memenuhi tugas mata kuliah Pembelajaran Mesin
2. Untuk menunjukkan dan menjelaskan hasil analisis dengan algoritma ANN
3. Untuk menunjukkan dan menjelaskan hasil akhir yang didapat dari dataset yang diberikan

STRUKTUR LAPORAN

1. Ikhtisar Kumpulan Data

   Data yang saya gunakan adalah arrhythmia (NIM Genap) dengan kriteria data sebagai berikut:

Data Set Characteristics:	Multivariate	**Number of Instances:**	452	**Area:**	Life
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	279	**Date Donated**	1998-01-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	410432

   Dataset arrhythmia bertujuan untuk menentukan jenis arrhythmia dan mengelompokkannya menjadi 16 kelas. Dataset ini terdiri dari 279 atribut dan 452 instansi, ditambah dengan satu atribut Class. Dataset ini juga memiliki missing value '?' pada sebagian data yang tersebar.

2. Ringkasan Pra-Pemrosesan Data

   Pada Pra-Pemrosesan data, saya melakukan berbagai tahap diantaranya eksplorasi data dan visualisasi data, untuk tahapannya sebagai berikut:
   Eksplorasi Data:
   Converting - Display Info - Counting Missing Data - Filling Data - Split & Scaling Data
   Visualisasi Data:
   Display Info - Data Distribution

3. Algoritma ANN

   Algoritma yang saya pilih untuk tugas ini adalah algoritma Artificial Neural Network. Karena Artificial Neural Network (ANN) yang dirasa paling cocok dan mudah untuk kasus ini, dimana ANN itu sendiri merupakan sebuah teknik untuk melakukan pengolahan informasi yang terinspirasi oleh cara kerja dari saraf biologis. Tujuan ANN yaitu menjadikan menjadikan komputer memiliki kemampuan kognitif yang sama dengan otak manusia yang memiliki banyak kemampuan seperti problem solving dan juga bisa melakukan pembelajaran.

4. Evaluasi Hasil

   Hasil evaluasi yang didapat adalah dengan keluaran akurasi yang didapatkan hanya berkisar 0 - 0.60, tidak sampai ke 0.70 dimana garis akurasi training nya lebih tinggi dibanding validationnya. Namun ada persamaan kestabilan akurasi ketika Epoch di rentang 0-20. Perbedaan garis akurasi tidak terlalu signifikan hal ini dikarenakan banyaknya data dan juga bisa jadi karena data yang digunakan adalah acak (random) jadi kemungkinan juga hasil akurasi dapat berbeda setiap data acaknya.

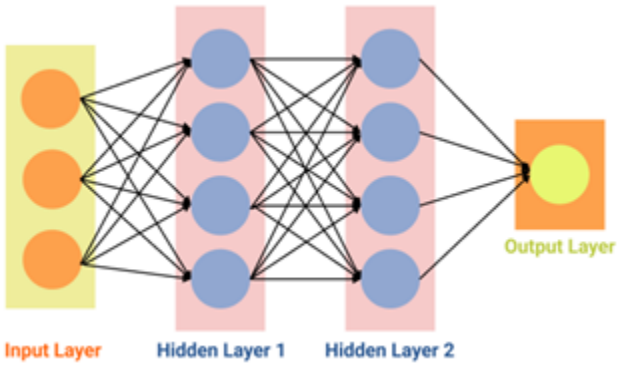
   (Lampiran Screenshot pada bagian IMPLEMENTASI PENGKODEAN)

PEMBAHASAN

A. *Artificial Neural Network*

Dalam melakukan pemodelan prediktif (prediksi), terdapat beberapa teknik yang paling populer digunakan, yaitu seperti *Decision Tree, Regression,* dan *Neural Network*. *Neural Network* merupakan model algoritma yang mencoba meniru otak manusia yang mampu memberikan stimulasi, melakukan proses, dan memberikan output untuk menemukan hubungan antara kumpulan data. *Neural Network* yang paling umum dikenal dengan *Artificial Neural Network* atau biasa disebut sebagai Jaringan Saraf Tiruan.

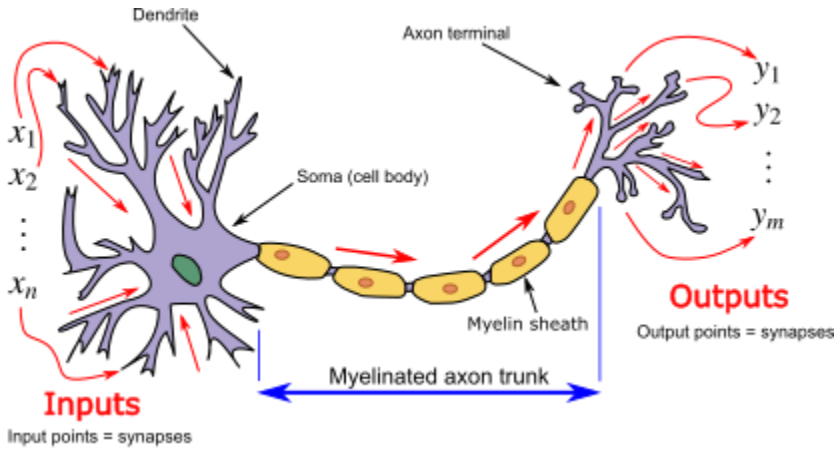
Artificial Neural Network merupakan salah satu pemodelan kompleks yang konsepnya dapat memprediksi bagaimana ekosistem merespon perubahan variabel lingkungan dengan terinspirasi oleh cara kerja sistem sel otak manusia dalam memproses informasi. Algoritma yang paling populer digunakan oleh ANN adalah *supervised learning.* Dikarenakan model ANN terinspirasi oleh sistem saraf biologis manusia, arsitekturnya juga dibuat seperti struktur otak manusia dimana terdiri dari neuron yang saling terhubung satu sama lain dan bentuk yang kompleks dan nonlinier. Berdasarkan penjelasan tersebut ANN dapat direpresentasikan menjadi 3 bagian yang terdiri dari layer masukan (*input*) dan layer keluar (*output*), dan layer tersembunyi (*hidden*) yang memproses masukan dari layer masukan menjadi sesuatu yang dapat diterima layer keluaran. Kemudian hasil dari masing-masing layer tersembunyi ini disebut *activation* atau *node value.*



1. Algoritma ANN

ANN terdiri dari neuron buatan yang secara konseptual berasal dari neuron biologis. Setiap neuron buatan memiliki *input* dan menghasilkan satu *output* yang dapat dikirim ke beberapa neuron lainnya. *Input* dapat berupa nilai fitur dari sampel data eksternal, seperti gambar atau dokumen, atau dapat berupa output dari neuron lain. *Output* dari neuron output akhir dari jaringan saraf menyelesaikan tugas, seperti mengenali objek dalam gambar.

Untuk menemukan *output* dari neuron kita mengambil jumlah bobot dari semua *input*, dibobot dengan bobot koneksi dari *input* ke neuron. Dapat menambahkan istilah bias untuk jumlah ini. Jumlah tertimbang ini sering disebut aktivasi. Jumlah tertimbang ini kemudian dilewatkan melalui fungsi aktivasi yang biasanya nonlinier untuk menghasilkan *output*. *Input* awal adalah data eksternal, seperti gambar dan dokumen. *Output* akhir menyelesaikan tugas, seperti mengenali objek dalam gambar.



2. Kelebihan Algoritma ANN

Terdapat kelebihan dari penggunaan algoritma ANN, berikut adalah kelebihan dari ANN.

a) Kemampuan pemrosesan paralel. ANN memiliki nilai numerik yang memungkinkannya untuk melakukan lebih dari satu tugas secara bersamaan.
b) Menyimpan data di seluruh jaringan. Alih-alih disimpan di *database*, data yang digunakan akan disimpan di seluruh jaringan. Dengan begitu, proses kerja jaringan tidak akan terhambat bila terjadi data hilang.
c) Menghasilkan output bahkan dengan data yang tidak memadai.
d) Memiliki distribusi memori.
e) Mampu mentoleransi kesalahan.

3. Kekurangan Algoritma ANN

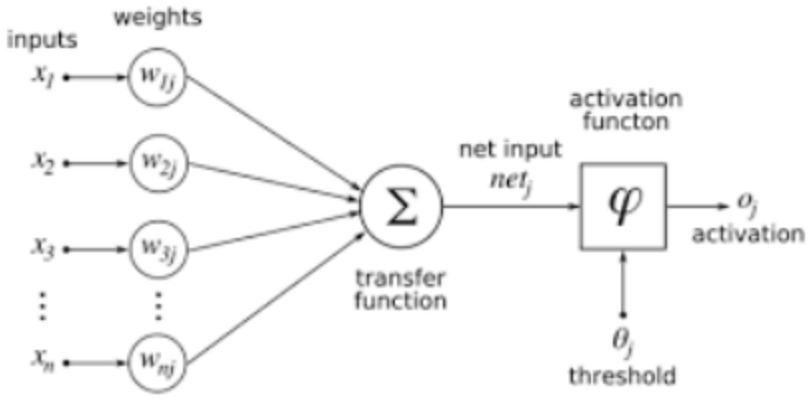
   Sementara itu, terdapat juga kekurangan dari algoritma ANN.

   a) Tidak ada pedoman khusus untuk menentukan struktur ANN. Anda perlu mengalami beberapa percobaan dan kegagalan agar berhasil.
   b) Perilaku jaringan yang tidak dikenali. Hal ini adalah kekurangan yang paling menonjol dari ANN. Ketika ANN menghasilkan solusi pengujian, tidak ada penjelasan tentang mengapa dan bagaimananya.
   c) Ketergantungan pada hardware. ANN memerlukan prosesor dengan kekuatan pemrosesan paralel sesuai dengan strukturnya.
   d) Sulit menyampaikan masalah ke jaringan. Hal ini akan bergantung pada kemampuan Anda sebagai pengguna.
   e) Durasi jaringan tidak diketahui. Jaringan akan direduksi ke nilai kesalahan tertentu dan nilai ini tidak memberikan hasil yang optimal.

4. Cara Kerja Algoritma ANN

   Secara umum, algoritma *Artificial Neural Network* bekerja dengan menerima input dalam berbagai format. Kemudian, masing-masing input dikalikan dengan bobot yang sesuai. Bobot adalah detail yang digunakan oleh ANN untuk memecahkan suatu masalah tertentu. Umumnya, bobot juga merujuk pada kekuatan interkoneksi antar-neuron pada jaringan saraf *Artificial*. Bila jumlah bobot sama dengan 0, maka perlu ditambahkan bias untuk membuat output tidak sama dengan 0. Bila bias memiliki input yang sama, maka bobot pun sama dengan 1. Total input berbobot dapat berada dalam kisaran 0 hingga tak terhingga positif. Untuk menjaga respons agar berada dalam batas nilai yang dimau, maka nilai maksimum tertentu harus di-benchmark. Di sisi lain, total input berbobot dilewatkan menggunakan fungsi aktivasi.

   Fungsi aktivasi pada Artificial Neural Networks mengacu pada set fungsi transfer yang dapat digunakan untuk mencapai output yang diinginkan. Beberapa himpunan fungsi yang biasanya digunakan adalah fungsi aktivasi sigmoid hiperbolik biner, linear, dan tan.



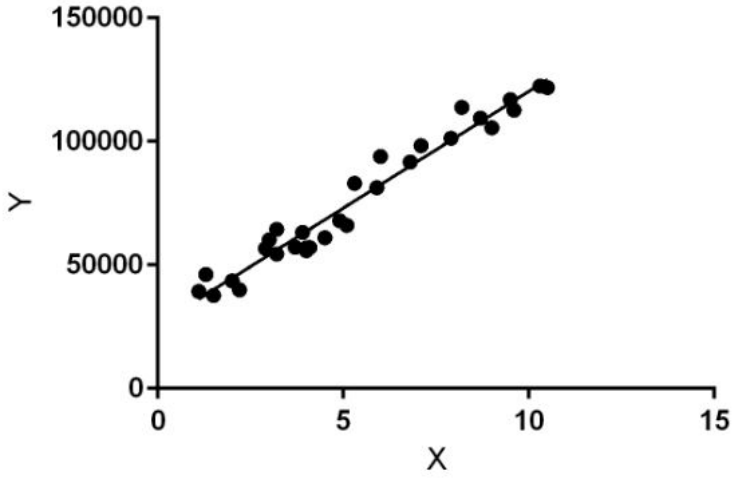
5. Linear Regression

Linear Regression adalah suatu regresi linear yang digunakan untuk mengestimasi atau memprediksi hubungan antara dua variabel dalam penelitian kuantitatif. Dimana regresi linear ini mampu membuat satu asumsi tambahan yang mengkorelasikan antara variabel independen dan dependen melalui garis yang paling sesuai dari titik data garis lurus. Meski demikian, regresi linear memiliki keterbatasan, karena dalam data terbaik pun tidak menceritakan kisah yang lengkap. Analisis regresi biasanya digunakan dalam penelitian untuk menetapkan bahwa ada korelasi antar variabel.

Namun, korelasi yang tidak sama sebab akibat adanya hubungan antara dua variabel tidak berarti yang satu menyebabkan yang lainnya terjadi. Bahkan garis dalam regresi linear sederhana yang cocok dengan titik data mungkin tidak menjamin hubungan sebab-akibat.

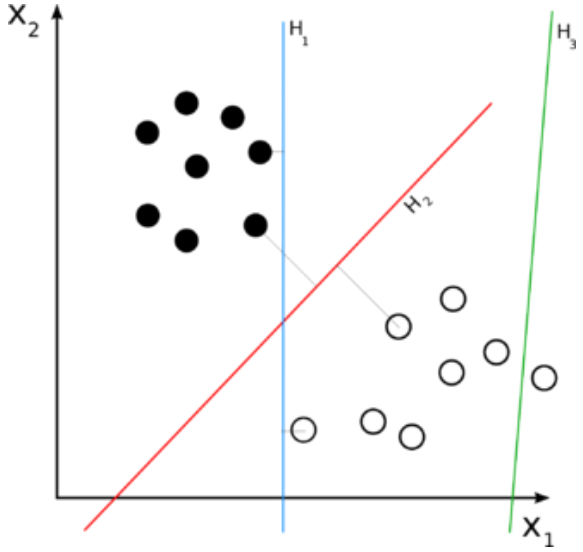
Ketika hubungan antara variabel independen dan dependen memiliki hubungan linear, algoritma ini adalah yang terbaik untuk digunakan, karena ini adalah yang paling kompleks dibanding algoritma lain yang juga menemukan hubungan antara variabel independen dan dependen. Metode ini mampu digunakan untuk memprediksi nilai yang ada pada masa depan. Hal ini sejalan dengan fungsi dari analisis regresi yang dapat digunakan untuk peramalan dan prediksi.

Pada kenyataannya, dalam data real, jarang masalah di dunia yang menunjukkan hubungan yang jelas antara variabel dependen dan independen. Hal ini dapat membuat model yang tidak cukup bagus, disebabkan karena kesalahan dalam memilih variabel yang digunakan untuk analisis.



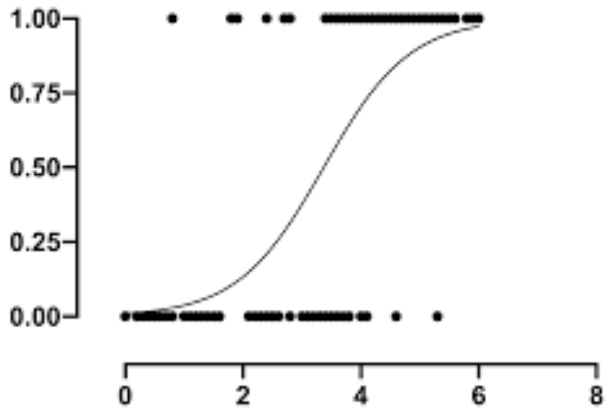
6. *Linear Classification*

Linear Classification adalah klasifikasi yang mengambil keputusan berdasarkan nilai kombinasi linear dari sifat-sifatnya. Sifat-sifat suatu objek disebut juga nilai fitur (ciri) dan biasa dinyatakan dalam vektor yang disebut vektor fitur. Pengklasifikasi ini bekerja dengan baik untuk permasalahan praktis seperti klasifikasi dokumen dan masalah-masalah yang memiliki banyak variabel (fitur) hingga mencapai tingkat akurasi yang sekelas dengan pengklasifikasi nonlinear dengan waktu pelatihan yang lebih sedikit.



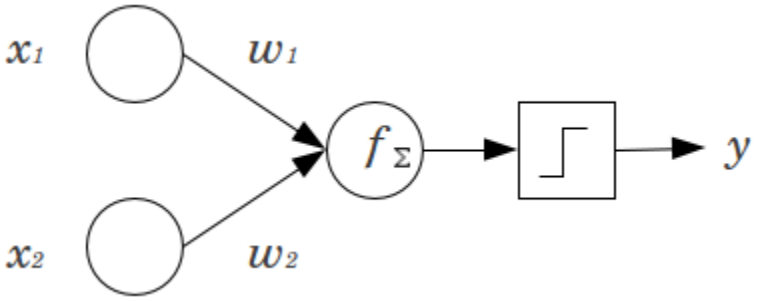
7. Logistic Regression

Logistic Regression adalah sebuah algoritma klasifikasi untuk mencari hubungan antara fitur (input) diskrit/kontinu dengan probabilitas hasil output diskrit tertentu. Tipe-tipe Logistic Regression adalah Binary Logistic Regression, Multinomial Logistic Regression, dan Ordinal Logistic Regression.



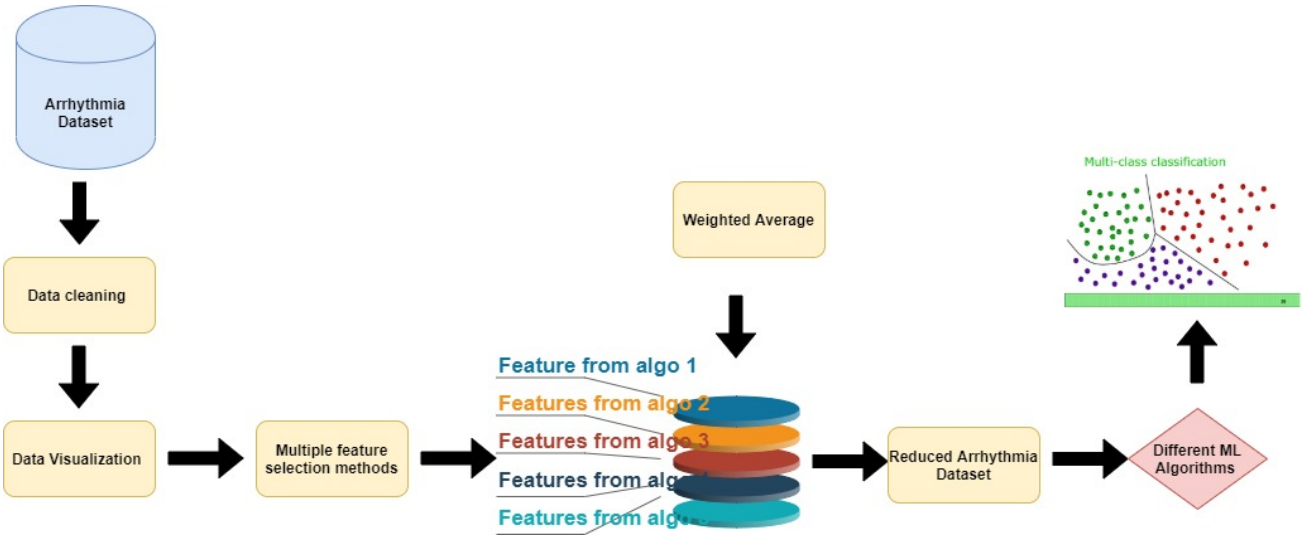
8. Perceptron

Perceptron adalah algoritma pada Supervised Learning untuk melakukan klasifikasi biner (dua jenis). Klasifikasi biner menggunakan suatu fungsi yang menentukan suatu data termasuk dalam suatu kelompok atau tidak. Ia termasuk jenis pengklasifikasi linear, yaitu algoritma klasifikasi yang memprediksi dengan fungsi linear yang tersusun dari himpunan bobot dengan vektor fitur.



9. *Arrhythmia Dataset*

Arrhythmia Dataset adalah data yang berisi tentang informasi lengkap (secara detail) penyakit Aritmia. Aritmia dianggap sebagai penyakit yang mengancam jiwa yang menyebabkan masalah kesehatan serius pada pasien, jika tidak diobati. Diagnosis dini aritmia akan sangat membantu dalam menyelamatkan nyawa. Penelitian ini dilakukan untuk mengklasifikasikan pasien ke dalam salah satu dari 16 subkelas, di antaranya 1 kelas mewakili tidak adanya penyakit dan 15 kelas lainnya mewakili catatan elektrokardiogram dari berbagai subtipe aritmia. Penelitian dilakukan pada dataset yang diambil dari University of California di Irvine Machine Learning Data Repository.



High-Level Overview :

- Menggunakan standard open source dataset dari UCI Repository
- Data cleaning
- Peningkatan Feature selection and Feature engineering
- Metode rata-rata tertimbang baru untuk mendapatkan fitur terbaik untuk mewakili informasi penting dalam data
- Metode klasifikasi Multiclass baru untuk mengalahkan akurasi canggih dan mencapai 94,2 dengan fitur penting secara medis dan 95,8 tanpa fitur penting secara medis
- Arsitektur hybrid dijalankan untuk memastikan lebih sedikit komputasi
- Ensemble fusion framework dari output semua model untuk memberikan wawasan yang lebih baik
- Menggunakan teknik bagging dan sampling yang berbeda untuk mendapatkan akurasi yang canggih

IMPLEMENTASI PENGKODEAN ALGORITMA

Untuk melakukan analisis dan evaluasi algoritma *Artificial Neural Networks*, berikut adalah tahapan implementasi untuk proses algoritma ANN yang telah saya buat sedemikian rupa dari tahap awal (*pre-processing*) sampai tahap akhir (analisis) dengan menggunakan bahasa pemrograman Python.

1. *Import Dataset*

```
[64] # https://drive.google.com/file/d/1YiKKjFouaVyz8mIMRxKiDGY0rbhL2Oo9/view?usp=sharing
     # https://drive.google.com/file/d/1pHuiTUZgppq_R8U8leEplkY_YaQTfia4/view?usp=sharing

     !gdown 1YiKKjFouaVyz8mIMRxKiDGY0rbhL2Oo9
     !gdown 1pHuiTUZgppq_R8U8leEplkY_YaQTfia4
```

```
Downloading...
From: https://drive.google.com/uc?id=1YiKKjFouaVyz8mIMRxKiDGY0rbhL2Oo9
To: /content/arrhythmia.data
100% 402k/402k [00:00<00:00, 101MB/s]
Downloading...
From: https://drive.google.com/uc?id=1pHuiTUZgppq_R8U8leEplkY_YaQTfia4
To: /content/arrhythmia.names
100% 6.16k/6.16k [00:00<00:00, 8.55MB/s]
```

```
[66] #uploading file dataset
     from google.colab import files
     uploaded = files.upload()
```

Choose Files 2 files

- **arrhythmia.data**(n/a) - 402355 bytes, last modified: 10/31/2022 - 100% done
- **arrhythmia.names**(n/a) - 6162 bytes, last modified: 10/31/2022 - 100% done

```
Saving arrhythmia.data to arrhythmia (2).data
Saving arrhythmia.names to arrhythmia (2).names
```

2. *Import Library*

```
#importing library untuk preprocessing dan visualisasi-exploring data

import pandas as pd

from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier

from sklearn.pipeline import Pipeline
from sklearn import metrics

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense
from mlxtend.plotting import plot_decision_regions

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
```

3. *Converting Dataset*

```
[67] #converting dataframe
dataArrhy = pd.read_csv("arrhythmia.data", header=None,
                        #na_values='?'
                        ) #mengubah missing data ke na_values(NaN)
dataArrhy.head()
```

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277	278	279
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

5 rows × 280 columns

```
dataArrhy.shape
```

```
(452, 280)
```

```
[70] dataArrhy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 452 entries, 0 to 451
Columns: 280 entries, 0 to 279
dtypes: float64(120), int64(155), object(5)
memory usage: 988.9+ KB
```

```
[71] dataArrhy.describe().transpose().head()
```

	count	mean	std	min	25%	50%	75%	max
0	452.0	46.471239	16.466631	0.0	36.0	47.0	58.0	83.0
1	452.0	0.550885	0.497955	0.0	0.0	1.0	1.0	1.0
2	452.0	166.188053	37.170340	105.0	160.0	164.0	170.0	780.0
3	452.0	68.170354	16.590803	6.0	59.0	68.0	79.0	176.0
4	452.0	88.920354	15.364394	55.0	80.0	86.0	94.0	188.0

4. *Counting Missing Data*

```
[72] #counting missing data
     count = 0
     for i in range(0,452):
         for j in range(0,280):
             if (dataArrhy.iloc[i,j] == '?'):
                 count = count + 1
     print(count)
     dataArrhy = dataArrhy.replace('?', np.NaN)
```

```
408
```

```
[73] dataArrhy.isnull().sum()
```

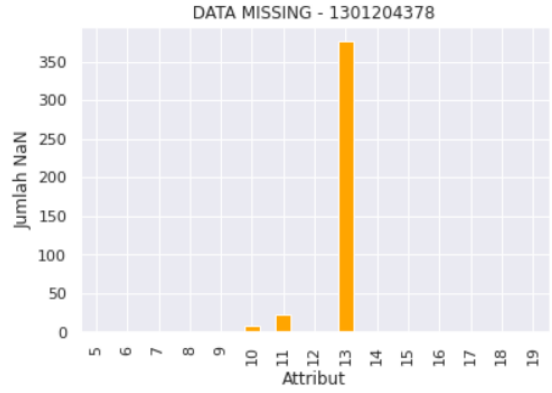
```
0      0
1      0
2      0
3      0
4      0
      ..
275    0
276    0
277    0
278    0
279    0
Length: 280, dtype: int64
```

```
#visualisasi dari penyebaran missing data
pd.isnull(dataArrhy).sum().plot(color='orange')
plt.title(" DATA MISSING - 1301204378 ")
plt.xlabel('Attribut')
plt.ylabel('Jumlah NaN')
```

```
Text(0, 0.5, 'Jumlah NaN')
```


```
[75] pd.isnull(dataArrhy).sum()[5:20].plot(kind='bar',color='orange')
     plt.title(" DATA MISSING - 1301204378 ")
     plt.xlabel('Attribut')
     plt.ylabel('Jumlah NaN')
```

```
Text(0, 0.5, 'Jumlah NaN')
```



5. Proses Pengolahan Data *Null*

```
[76] #dropping column 13
     dataArrhy.drop(columns = 13, inplace=True)
```

```
[77] dataArrhy = pd.DataFrame(dataArrhy)
     dataArrhy.head()
```

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277	278	279
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

5 rows × 279 columns

```
[ ] dataArrhy.isnull().sum()
```

```
0      0
1      0
2      0
3      0
4      0
      ..
275    0
276    0
277    0
278    0
279    0
Length: 279, dtype: int64
```

```
[79] #checking missing data yang tersisa
     dataArrhy.isna().sum().sum()
```

```
32
```

```
[80] #replacing missing data
     dataArrhy = dataArrhy.replace("?", np.nan)
     dataArrhy.isin(["?"]).sum().sum()
```

```
0
```

```
[81] #filling column missing data dengan rata-rata kolom
     dataArrhy = dataArrhy.fillna(dataArrhy.mean())
     dataArrhy
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in Dat
```

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277	278	279
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
447	53	1	160	70	80	199	382	154	117	-37	...	0.0	4.3	-5.0	0.0	0.0	0.7	0.6	-4.4	-0.5	1
448	37	0	190	85	100	137	361	201	73	86	...	0.0	15.6	-1.6	0.0	0.0	0.4	2.4	38.0	62.4	10
449	36	0	166	68	108	176	365	194	116	-85	...	0.0	16.3	-28.6	0.0	0.0	1.5	1.0	-44.2	-33.2	2
450	32	1	155	55	93	106	386	218	63	54	...	-0.4	12.0	-0.7	0.0	0.0	0.5	2.4	25.0	46.6	1
451	78	1	160	70	79	127	364	138	78	28	...	0.0	10.4	-1.8	0.0	0.0	0.5	1.6	21.3	32.8	1

452 rows × 279 columns

6. Proses *Converting* Atribut *Objective* ke *Numeric*

```
#replacing OBJECT to NUMERIC
kolom = dataArrhy.columns[dataArrhy.dtypes.eq("object")]
dataArrhy[kolom] = dataArrhy[kolom].apply(pd.to_numeric, errors="coerce")
dataArrhy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 452 entries, 0 to 451
Columns: 279 entries, 0 to 279
dtypes: float64(124), int64(155)
memory usage: 985.3 KB
```

7. *Filling Name Column*

```
#filling nama column dengan manual dari variabel dataFix
dataFix_columns = [ "Age","Sex","Height","Weight","QRS_Dur", "P-R_Int","Q-T_Int",
"T_Int","P_Int","QRS","T","P","J","Heart_Rate","Q_Wave","R_Wave","S_Wave","R'_Wave","S'_Wave",
"Int_Def","Rag_R_Nom","Diph_R_Nom","Rag_P_Nom","Diph_P_Nom","Rag_T_Nom","Diph_T_Nom",
"DII00", "DII01","DII02", "DII03", "DII04","DII05","DII06","DII07","DII08","DII09","DII10","DII11",
"DIII00","DIII01","DIII02", "DIII03", "DIII04","DIII05","DIII06","DIII07","DIII08","DIII09","DIII10","DIII11",
"AVR00","AVR01","AVR02","AVR03","AVR04","AVR05","AVR06","AVR07","AVR08","AVR09","AVR10","AVR11",
"AVL00","AVL01","AVL02","AVL03","AVL04","AVL05","AVL06","AVL07","AVL08","AVL09","AVL10","AVL11",
"AVF00","AVF01","AVF02","AVF03","AVF04","AVF05","AVF06","AVF07","AVF08","AVF09","AVF10","AVF11",
"V100","V101","V102","V103","V104","V105","V106","V107","V108","V109","V110","V111",
"V200","V201","V202","V203","V204","V205","V206","V207","V208","V209","V210","V211",
"V300","V301","V302","V303","V304","V305","V306","V307","V308","V309","V310","V311",
"V400","V401","V402","V403","V404","V405","V406","V407","V408","V409","V410","V411",
"V500","V501","V502","V503","V504","V505","V506","V507","V508","V509","V510","V511",
"V600","V601","V602","V603","V604","V605","V606","V607","V608","V609","V610","V611",
"JJ_Wave","Amp_Q_Wave","Amp_R_Wave","Amp_S_Wave","R_Prime_Wave","S_Prime_Wave","P_Wave","T_Wave",
"QRSA","QRSTA","DII170","DII171","DII172","DII173","DII174","DII175","DII176","DII177","DII178","DII179",
"DIII180","DIII181","DIII182","DIII183","DIII184","DIII185","DIII186","DIII187","DIII188","DIII189",
"AVR190","AVR191","AVR192","AVR193","AVR194","AVR195","AVR196","AVR197","AVR198","AVR199",
"AVL200","AVL201","AVL202","AVL203","AVL204","AVL205","AVL206","AVL207","AVL208","AVL209",
"AVF210","AVF211","AVF212","AVF213","AVF214","AVF215","AVF216","AVF217","AVF218","AVF219",
"V1220","V1221","V1222","V1223","V1224","V1225","V1226","V1227","V1228","V1229",
"V2230","V2231","V2232","V2233","V2234","V2235","V2236","V2237","V2238","V2239",
"V3240","V3241","V3242","V3243","V3244","V3245","V3246","V3247","V3248","V3249",
"V4250","V4251","V4252","V4253","V4254","V4255","V4256","V4257","V4258","V4259",
"V5260","V5261","V5262","V5263","V5264","V5265","V5266","V5267","V5268","V5269",
"V6270","V6271","V6272","V6273","V6274","V6275","V6276","V6277","V6278","V6279",
"ClassCode"]
```

```
[84] #Deklarasi kolom yang sudah ditambahkan
     dataArrhy.columns = dataFix_columns
     dataArrhy
```

```
[85] dataArrhyColumns = dataArrhy.columns.values
     print('Columns in Arrhythmia Dataset\n {}'.format(dataArrhyColumns))
```

```
Columns in Arrhythmia Dataset
 ['Age' 'Sex' 'Height' 'Weight' 'QRS_Dur' 'P-R_Int' 'Q-T_Int' 'T_Int'
 'P_Int' 'QRS' 'T' 'P' 'J' 'Heart_Rate' 'Q_Wave' 'R_Wave' 'S_Wave'
 "R'_Wave" "S'_Wave" 'Int_Def' 'Rag_R_Nom' 'Diph_R_Nom' 'Rag_P_Nom'
 'Diph_P_Nom' 'Rag_T_Nom' 'Diph_T_Nom' 'DII00' 'DII01' 'DII02' 'DII03'
 'DII04' 'DII05' 'DII06' 'DII07' 'DII08' 'DII09' 'DII10' 'DII11' 'DIII00'
 'DIII01' 'DIII02' 'DIII03' 'DIII04' 'DIII05' 'DIII06' 'DIII07' 'DIII08'
 'DIII09' 'DIII10' 'DIII11' 'AVR00' 'AVR01' 'AVR02' 'AVR03' 'AVR04'
 'AVR05' 'AVR06' 'AVR07' 'AVR08' 'AVR09' 'AVR10' 'AVR11' 'AVL00' 'AVL01'
 'AVL02' 'AVL03' 'AVL04' 'AVL05' 'AVL06' 'AVL07' 'AVL08' 'AVL09' 'AVL10'
 'AVL11' 'AVF00' 'AVF01' 'AVF02' 'AVF03' 'AVF04' 'AVF05' 'AVF06' 'AVF07'
 'AVF08' 'AVF09' 'AVF10' 'AVF11' 'V100' 'V101' 'V102' 'V103' 'V104' 'V105'
 'V106' 'V107' 'V108' 'V109' 'V110' 'V111' 'V200' 'V201' 'V202' 'V203'
 'V204' 'V205' 'V206' 'V207' 'V208' 'V209' 'V210' 'V211' 'V300' 'V301'
 'V302' 'V303' 'V304' 'V305' 'V306' 'V307' 'V308' 'V309' 'V310' 'V311'
 'V400' 'V401' 'V402' 'V403' 'V404' 'V405' 'V406' 'V407' 'V408' 'V409'
 'V410' 'V411' 'V500' 'V501' 'V502' 'V503' 'V504' 'V505' 'V506' 'V507'
 'V508' 'V509' 'V510' 'V511' 'V600' 'V601' 'V602' 'V603' 'V604' 'V605'
 'V606' 'V607' 'V608' 'V609' 'V610' 'V611' 'JJ_Wave' 'Amp_Q_Wave'
 'Amp_R_Wave' 'Amp_S_Wave' 'R_Prime_Wave' 'S_Prime_Wave' 'P_Wave' 'T_Wave'
 'QRSA' 'QRSTA' 'DII170' 'DII171' 'DII172' 'DII173' 'DII174' 'DII175'
 'DII176' 'DII177' 'DII178' 'DII179' 'DIII180' 'DIII181' 'DIII182'
 'DIII183' 'DIII184' 'DIII185' 'DIII186' 'DIII187' 'DIII188' 'DIII189'
 'AVR190' 'AVR191' 'AVR192' 'AVR193' 'AVR194' 'AVR195' 'AVR196' 'AVR197'
 'AVR198' 'AVR199' 'AVL200' 'AVL201' 'AVL202' 'AVL203' 'AVL204' 'AVL205'
 'AVL206' 'AVL207' 'AVL208' 'AVL209' 'AVF210' 'AVF211' 'AVF212' 'AVF213'
 'AVF214' 'AVF215' 'AVF216' 'AVF217' 'AVF218' 'AVF219' 'V1220' 'V1221'
```

```
[86] #showing setelah ditambahkan dataFix
     dataArrhy.describe().transpose().head()
```

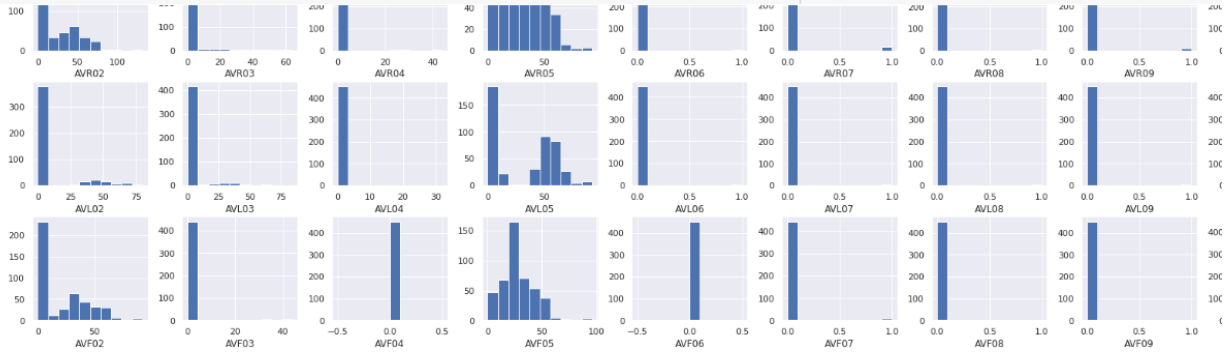
	count	mean	std	min	25%	50%	75%	max
Age	452.0	46.471239	16.466631	0.0	36.0	47.0	58.0	83.0
Sex	452.0	0.550885	0.497955	0.0	0.0	1.0	1.0	1.0
Height	452.0	166.188053	37.170340	105.0	160.0	164.0	170.0	780.0
Weight	452.0	68.170354	16.590803	6.0	59.0	68.0	79.0	176.0
QRS_Dur	452.0	88.920354	15.364394	55.0	80.0	86.0	94.0	188.0

```
[88] dataArrhy.info()
```

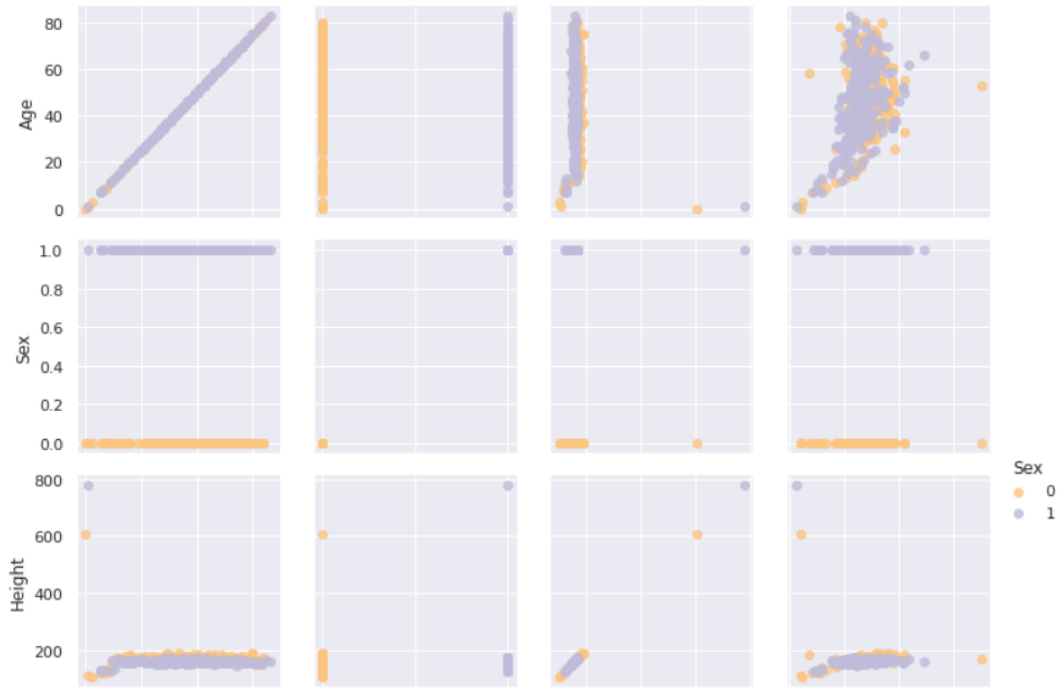
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 452 entries, 0 to 451
Columns: 279 entries, Age to ClassCode
dtypes: float64(124), int64(155)
memory usage: 985.3 KB
```

8. Visualisasi Data

```
[89] #Histogram distribusi data
     dataArrhy[dataArrhy.dtypes[(dataArrhy.dtypes=="float")].index.values].hist(figsize=[40,40])
     dataArrhy[dataArrhy.dtypes[(dataArrhy.dtypes=="int64")].index.values].hist(figsize=[40,40])
```



```
#visualisasi persebaran data #1
g = sns.PairGrid(dataArrhy, vars=['Age', 'Sex', 'Height', 'Weight'], #data manusia
                 hue='Sex', palette='PuOr')
g.map(plt.scatter, alpha=0.8)
g.add_legend();
```




```
[91] #sorting untuk atribut height (descending)
     sorted(dataArrhy['Height'], reverse=True)[:10]
```

```
[780, 608, 190, 190, 190, 188, 186, 186, 186, 185]
```

Karena tinggi manusia normal tidak ada yang sampai 780 / 608

Saya asumsikan dan mengganti 780 dan 608 dengan 180cm dan 108cm

---

Untuk atribut lain bisa diexplore dengan data.unique (selain height, nilai atribut normal)

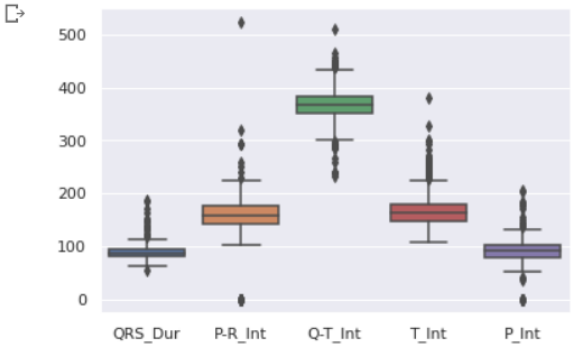
```
[92] #replacing height yang tidak normal
     dataArrhy['Height']=dataArrhy['Height'].replace(608,108)
     dataArrhy['Height']=dataArrhy['Height'].replace(780,180)

     sorted(dataArrhy['Weight'], reverse=True)[:10]
```

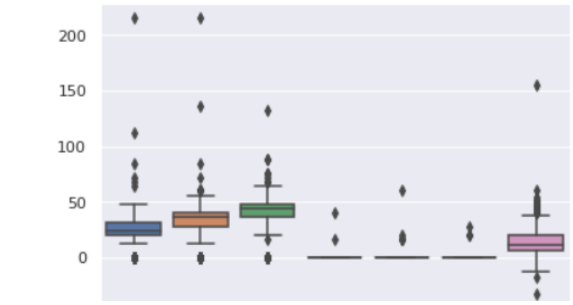
```
[176, 124, 110, 106, 105, 105, 104, 104, 100, 98]
```

Asumsikan berat badan 176 normal

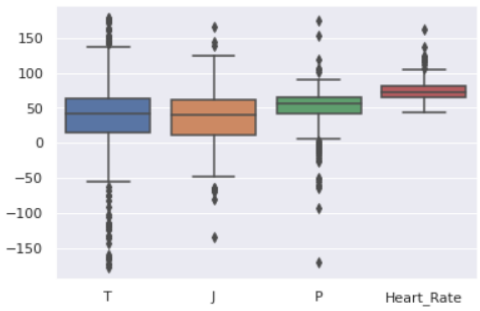
```
#visualisasi persebaran data #2
sns.boxplot(data=dataArrhy[["QRS_Dur","P-R_Int","Q-T_Int","T_Int","P_Int"]]);
```




```
[95] #visualisasi persebaran data #3
sns.boxplot(data=dataArrhy[["V101", "V201", "V301", "V404", "V503", "V603", "QRSA"]]);
```




```
#visualisasi persebaran data #4
sns.boxplot(data=dataArrhy[["T", "J", "P", "Heart_Rate"]]);
```



Hasil visualisasi diatas, terdapat pencilan data pada PR Interval, V101, V201, V301, V501 dan T, J, P, Heart_Rate. Karena data yang digunakan sangat bias, Saya asumsikan untuk tidak menghapus data pencilan tersebut dan memilih untuk tetap disimpan.

Cek data atribut mana yang masih terdapat pencilan

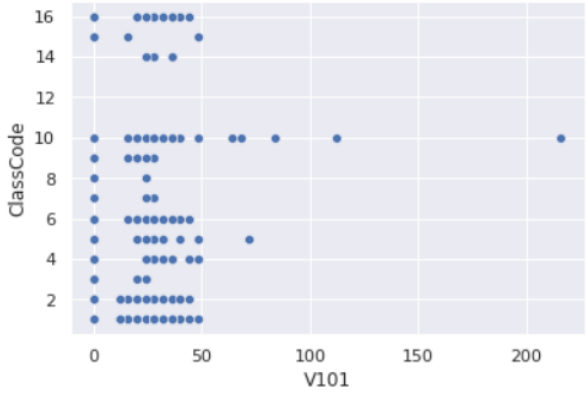
```
#mencari tahu data pencilan pada V101
dataArrhy["V101"].value_counts().sort_index(ascending=False)
```

```
216     1
112     1
84      1
72      1
68      1
64      1
48      6
44      6
40     13
36     36
32     63
28     81
24     88
20     57
16     13
12      4
0      79
Name: V101, dtype: int64
```

9. *Checking* dan *Filling* Data (*Mean, Median, Drop*)

```
#Checking persebaran data #1
sns.scatterplot(x='V101',y='ClassCode',data=dataArrhy)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3ce1aab250>
```

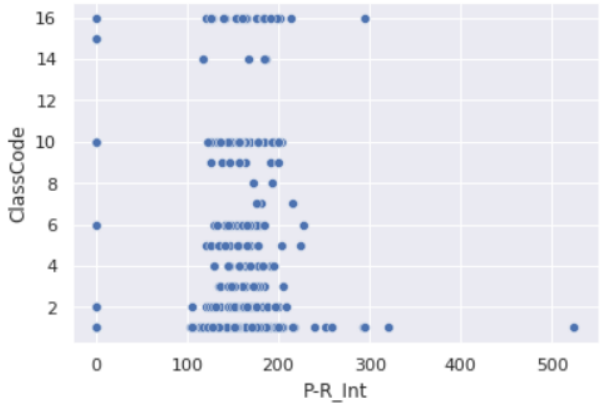


Tidak banyak data pencilan di kolom V101. Jadi kita menggunakan mean untuk mengisi nilai null.

```
[99] #filling kolom V101 dengan mean
     dataArrhy['V101']=dataArrhy['V101'].fillna(dataArrhy['V101'].mean())
```

```
#Checking persebaran data #2
sns.scatterplot(x='P-R_Int',y='ClassCode',data=dataArrhy)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3ce5a71690>
```



Ada banyak data pencilan di kolom ini. Kita menggunakan median untuk mengisi nilai nol.

```
[101] #filling kolom V101 dengan median
      dataArrhy['P-R_Int']=dataArrhy['P-R_Int'].fillna(dataArrhy['P-R_Int'].median())
```


```
[101] #filling kolom V101 dengan median
      dataArrhy['P-R_Int']=dataArrhy['P-R_Int'].fillna(dataArrhy['P-R_Int'].median())
```


Begitu Juga untuk Data yang lain (PR Interval, V101, V201, V301, V501 dan T, J, P, Heart_Rate)

Dengan Kriteria:

Jika atribut tersebut memiliki sedikit pencilan => Mean

Jika atribut tersebut memiliki banyak pencilan => Median

Jika atribut tersebut memiliki sangat banyak pencilan => DROP

```
[102] #filling kolom T dengan median
      dataArrhy['T']=dataArrhy['T'].fillna(dataArrhy['T'].median())
```

```
[103] #filling kolom J dengan mean
      dataArrhy['J']=dataArrhy['J'].fillna(dataArrhy['J'].mean())
```

```
[104] #filling kolom Heart_Rate dengan mean
      dataArrhy['Heart_Rate']=dataArrhy['Heart_Rate'].fillna(dataArrhy['Heart_Rate'].mean())
```

```
[105] dataArrhy.drop(['P'], inplace=True, axis=1)
```

```
[106] dataArrhy.isnull().sum().sum()
```

```
0
```

```
[107] dataArrhy.isnull().sum().head(20)
```

```
Age           0
Sex           0
Height        0
Weight        0
QRS_Dur       0
P-R_Int       0
Q-T_Int       0
T_Int         0
P_Int         0
QRS           0
T             0
J             0
Heart_Rate    0
Q_Wave        0
R_Wave        0
S_Wave        0
R'_Wave       0
S'_Wave       0
Int_Def       0
Rag_R_Nom     0
dtype: int64
```

10. Proses ANN

```
#Memisahkan data menjadi data training dan data testing
y = dataArrhy["ClassCode"]
x = dataArrhy.drop(["ClassCode"], axis=1)
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, test_size=0.2, random_state=3)
```

Saya mencoba melihat ukuran dari error data test yang akan dicoba

```
[113] #Fungsi untuk measure error
def score_dataset(y_test, x_test, pipeline):
  df_preds = pd.DataFrame((map(round, pipeline.predict(x_test))))
  y_test.index = range(0, len(y_test))
  df_reals = pd.DataFrame(y_test)
  df_reals = df_reals.values.tolist()
  df_preds = df_preds.values.tolist()

  count = 0
  for i in range(len(df_preds)):
    if df_preds[i] == df_reals[i]:
      count += 1
  score = (count/len(y_test))*100
  return score
```

```
[118] #Checking error (kesalahan)
      scores = ['n_estimate', 'error in %']
      df_score = pd.DataFrame(index=range(1, 280), columns=scores)
      for i in range(1, 280):
        my_pipeline = Pipeline(steps=[('preprocessor', SimpleImputer(strategy='constant', fill_value=0.0)), ('model', RandomForestClassifier(n_estimators=i, random_state=0))])
        my_pipeline.fit(x_train, y_train)
        df_score.loc[i, 'n_estimate'] = i
        df_score.loc[i, 'error in %'] = score_dataset(y_test, x_test, my_pipeline)
```

```
[119] df_score
```

	n_estimate	error in %
1	1	63.736264
2	2	67.032967
3	3	65.934066
4	4	67.032967
5	5	70.32967
...	...	...
275	275	72.527473
276	276	72.527473
277	277	72.527473
278	278	72.527473
279	279	72.527473

11. *Data Scaling* (MIN-MAX Scaling)

```
[120] #minmax scalling
      scaler = MinMaxScaler()

      x_train = scaler.fit_transform(x_train)
      x_test  = scaler.transform(x_test)
```

12. Splitting Data

```
#Split kolom ClassCode
y = dataArrhy["ClassCode"]
x = dataArrhy.drop(["ClassCode"], axis=1)
y = np.array(y)
x = np.array(x)
x [:10], y[:10]
```

```
(array([[ 75. ,   0. , 190. , ...,   2.9,  23.3,  49.4],
        [ 56. ,   1. , 165. , ...,   2.1,  20.4,  38.8],
        [ 54. ,   0. , 172. , ...,   3.4,  12.3,  49. ],
        ...,
        [ 49. ,   1. , 162. , ...,   0.5,  15.8,  19.8],
        [ 44. ,   0. , 168. , ...,   2.1,  12.5,  30.9],
        [ 50. ,   1. , 167. , ...,   0.9,  20.1,  25.1]])
 array([ 8,  6, 10,  1,  7, 14,  1,  1,  1, 10]))
```

13. Encoding Data

```
#Encoding  kolom classcode
l_encode = LabelEncoder()
l_encode.fit(y)
y = l_encode.transform(y)
y = to_categorical(y)
```

```
[137] #memisah dataset menjadi 2
x_train, x_test, y_train, y_test =  train_test_split(x,y, test_size = 0.3, random_state = 0)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((316, 277), (316, 13), (136, 277), (136, 13))
```

14. Processing

```
[138] dataArrhy_len = len(dataArrhy.columns)-1

      model = Sequential()
      model.add(Dense(452, input_dim = dataArrhy_len, activation = 'sigmoid'))
      model.add(Dense(452, activation = 'sigmoid'))
      model.add(Dense(452, activation = 'sigmoid'))
      model.add(Dense(452, activation = 'sigmoid'))
      model.add(Dense(452, activation = 'sigmoid'))
      model.add(Dense(452, activation = 'sigmoid'))
      model.add(Dense(452, activation = 'sigmoid'))
      model.add(Dense(452, activation = 'sigmoid'))
      model.add(Dense(13, activation = 'softmax'))

      model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
      hist = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=45)
      acc = hist.history['accuracy']
      val = hist.history['val_accuracy']
      epochs = range(1, len(acc) + 1)

      plt.plot(epochs, acc, '-', label='Training accuracy')
      plt.plot(epochs, val, ':', label='Validation accuracy')
      plt.title('Training and Validation Accuracy')
      plt.xlabel('Epoch')
      plt.ylabel('Accuracy')
      plt.legend(loc='lower right')
      plt.plot()
```

```
Epoch 29/45
10/10 [==============================] - 0s 34ms/step - loss: 1.3000 - accuracy: 0.6076 - val_loss: 1.6524 - val_accuracy: 0.5588
Epoch 30/45
10/10 [==============================] - 0s 30ms/step - loss: 1.3114 - accuracy: 0.5918 - val_loss: 1.5892 - val_accuracy: 0.5809
Epoch 31/45
10/10 [==============================] - 0s 34ms/step - loss: 1.2974 - accuracy: 0.6108 - val_loss: 1.5851 - val_accuracy: 0.5368
Epoch 32/45
10/10 [==============================] - 0s 32ms/step - loss: 1.2748 - accuracy: 0.6139 - val_loss: 1.7358 - val_accuracy: 0.5074
Epoch 33/45
10/10 [==============================] - 0s 33ms/step - loss: 1.2799 - accuracy: 0.5728 - val_loss: 1.5875 - val_accuracy: 0.5441
Epoch 34/45
10/10 [==============================] - 0s 36ms/step - loss: 1.2460 - accuracy: 0.6044 - val_loss: 1.6117 - val_accuracy: 0.5735
Epoch 35/45
10/10 [==============================] - 0s 38ms/step - loss: 1.2091 - accuracy: 0.6076 - val_loss: 1.5150 - val_accuracy: 0.5809
Epoch 36/45
10/10 [==============================] - 0s 38ms/step - loss: 1.2408 - accuracy: 0.6108 - val_loss: 1.9338 - val_accuracy: 0.5441
Epoch 37/45
10/10 [==============================] - 0s 36ms/step - loss: 1.2831 - accuracy: 0.6013 - val_loss: 1.6013 - val_accuracy: 0.5662
Epoch 38/45
10/10 [==============================] - 0s 41ms/step - loss: 1.1858 - accuracy: 0.6329 - val_loss: 1.6091 - val_accuracy: 0.5882
Epoch 39/45
10/10 [==============================] - 0s 36ms/step - loss: 1.1751 - accuracy: 0.6234 - val_loss: 1.6372 - val_accuracy: 0.5441
Epoch 40/45
10/10 [==============================] - 0s 33ms/step - loss: 1.1574 - accuracy: 0.6297 - val_loss: 1.5712 - val_accuracy: 0.5441
Epoch 41/45
10/10 [==============================] - 0s 33ms/step - loss: 1.1920 - accuracy: 0.6329 - val_loss: 2.0655 - val_accuracy: 0.4118
Epoch 42/45
10/10 [==============================] - 0s 33ms/step - loss: 1.3561 - accuracy: 0.5854 - val_loss: 1.7684 - val_accuracy: 0.5000
Epoch 43/45
10/10 [==============================] - 0s 35ms/step - loss: 1.2948 - accuracy: 0.6234 - val_loss: 1.5838 - val_accuracy: 0.5441
Epoch 44/45
10/10 [==============================] - 0s 36ms/step - loss: 1.2090 - accuracy: 0.6044 - val_loss: 1.4989 - val_accuracy: 0.6029
Epoch 45/45
10/10 [==============================] - 1s 54ms/step - loss: 1.1453 - accuracy: 0.6361 - val_loss: 1.5468 - val_accuracy: 0.5588
[]
```

15. Result dan *Prediction*

```
[139] results = model.evaluate(x_test, y_test, batch_size=128)
```

```
2/2 [==============================] - 0s 13ms/step - loss: 1.5468 - accuracy: 0.5588
```

```
[140] prediction = model.predict(x_test)
```

```
5/5 [==============================] - 0s 11ms/step
```

```
[142] y_test
```

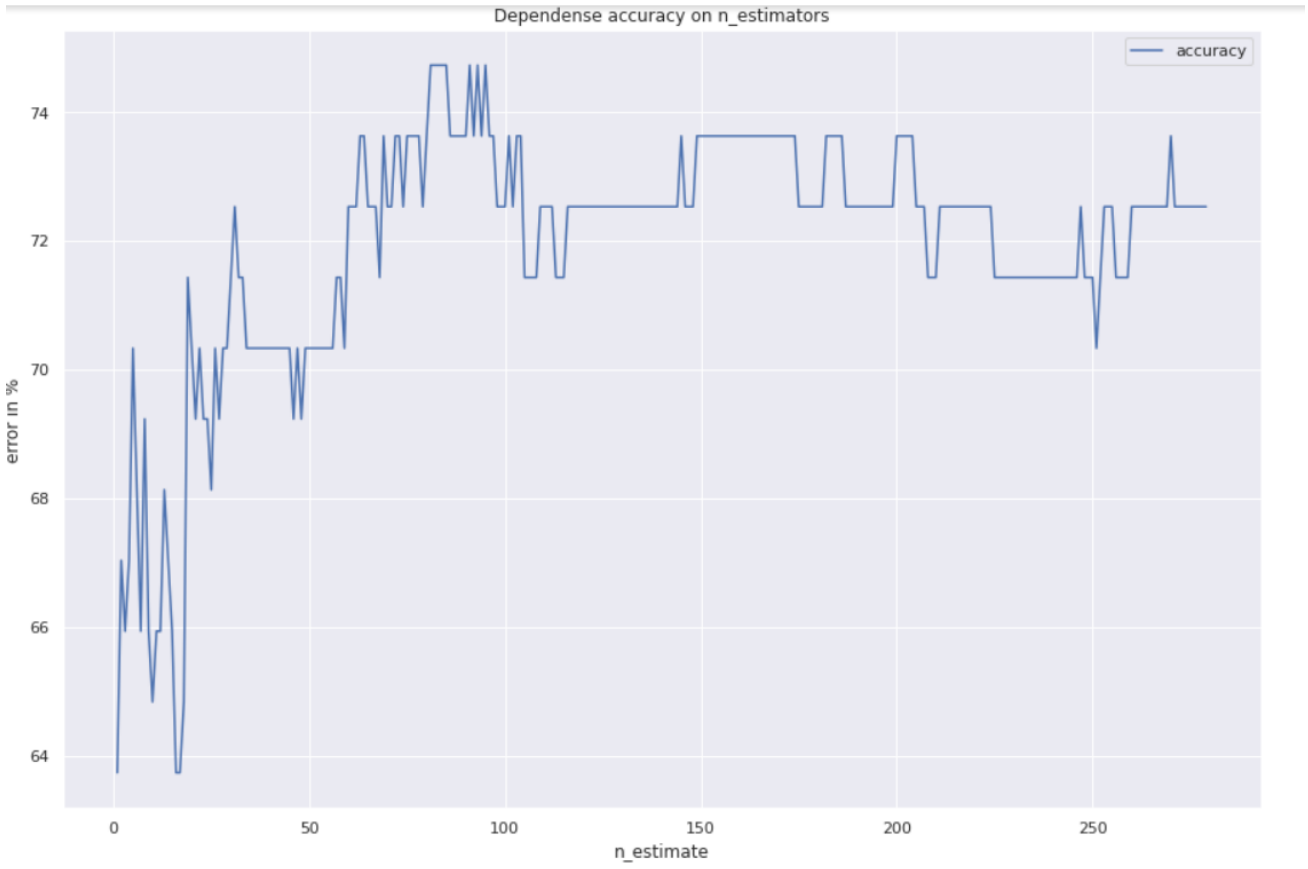
```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.],
       [1., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

16. Dependency Akurasi

```
[143] #Dependense dari akurasi #1
      df_score['n_estimate'] = df_score['n_estimate'].astype(int)
      df_score['error in %'] = df_score['error in %'].astype(float)

      plt.figure(figsize=(15, 10))
      sns.lineplot(x=df_score['n_estimate'], y=df_score['error in %'], label='accuracy')
      plt.title('Dependense accuracy on n_estimators')
```

```
Text(0.5, 1.0, 'Dependense accuracy on n_estimators')
```



ANALISIS DAN EVALUASI

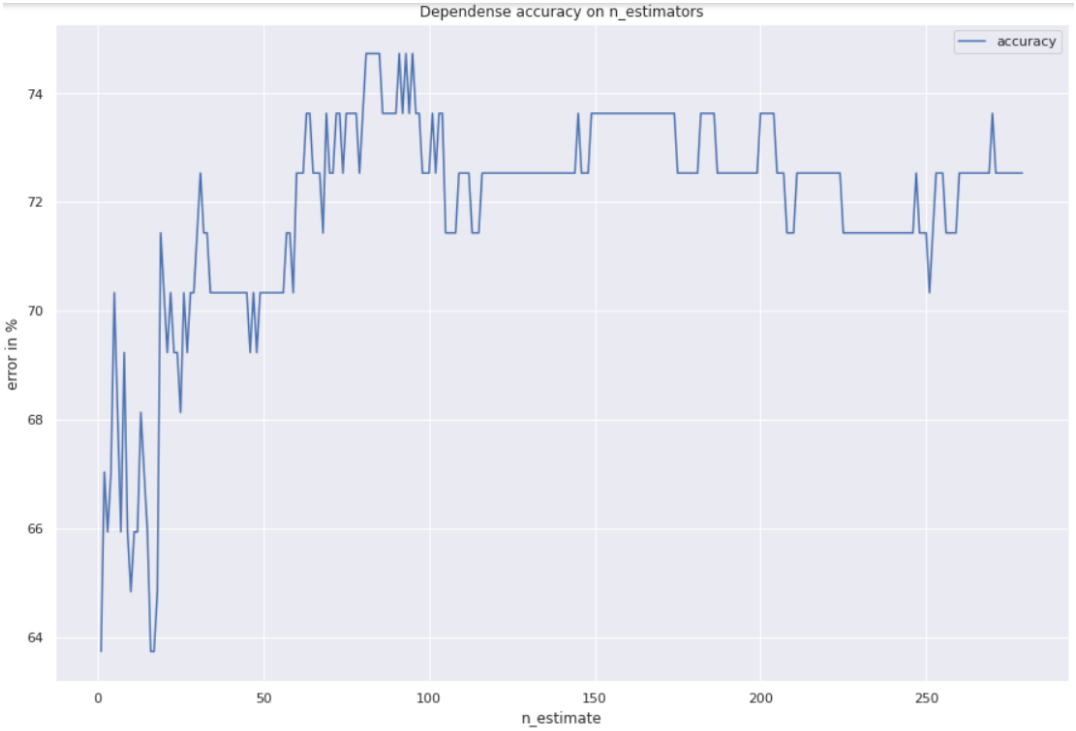
A. Analisis dan Evaluasi

Berikut merupakan analisis dan evaluasi dari proses implementasi algoritma *Artificial Neural Network* yang telah saya buat.

- Akurasi Training dan Validation

Akurasi yang didapatkan diatas hanya berkisar 0 - 0.60, tidak sampai ke 0.70 dimana garis akurasi training nya lebih tinggi dibanding validationnya. Namun ada persamaan kestabilan akurasi ketika Epoch di rentang 0-20. Perbedaan garis akurasi tidak terlalu signifikan hal ini dikarenakan banyaknya data dan juga bisa jadi karena data yang digunakan adalah acak (random) jadi kemungkinan juga hasil akurasi dapat berbeda setiap data acaknya.

- Akurasi Dependasi pada Estimator



Hasil akurasi diatas tidak bagus. Nilai puncak akurasi sebesar 73% dengan n_estimate dari 125 hingga 150. Mungkin, ini disebabkan nilai-nilai dari beberapa kelas sama sekali tidak ada, dan yang lain disajikan dalam sejumlah kecil nilai, sehingga beberapa dari mereka mungkin tidak masuk data training atau data testing. n_estimate yang lebih besar juga tidak menunjukkan hasil yang baik.

PENUTUP

A. Kesimpulan

ANN adalah singkatan dari *Artificial Neural Network* yang merupakan cabang dari Artificial Intelligence yang mengubah pola terhubung menjadi sekumpulan data, kemudian diproses kembali oleh komputer. *Supervised Learning* adalah salah satu contoh algoritma yang popular digunakan oleh ANN. Tujuan dari ANN adalah menjadikan suatu komputer yang memiliki skill seperti otak manusia, skill untuk problem solving dan dapat melakukan proses learning.

Artificial Neural Network ini memiliki kemampuan yang luar biasa untuk mendapatkan informasi dari data yang rumit atau tidak tepat sehingga permasalahan yang tidak terstruktur dan sulit didefinisikan dapat diatasi. Selain itu, ANN juga memiliki kemampuan untuk melakukan perhitungan secara paralel sehingga proses komputasi menjadi lebih cepat dan dapat membangun representasi dari informasi yang diterimanya selama proses pembelajaran secara mandiri. Namun, *Artificial Neural Network* memiliki kelemahan dimana tidak efektif jika digunakan untuk melakukan operasi-operasi numerik dengan presisi tinggi dan membutuhkan pelatihan dalam waktu yang lama jika jumlah data yang diolah besar.

B. Daftar Pustaka

Huang, Y. (2009). Advances in Artificial Neural Networks – Methodological Development and Application. Algorithms, 2(3), 973–1007. https://doi.org/10.3390/algor2030973

GeeksforGeeks. (2021, October 19). Introduction to Artificial Neutral Networks | Set 1. https://www.geeksforgeeks.org/introduction-to-artificial-neutral-networks/

GeeksforGeeks. (2021a, August 31). Artificial Neural Networks and its Applications. https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

Education, I. C. (2021, August 3). Neural Networks. https://www.ibm.com/cloud/learn/neural-networks

Karayiannis, N., & Venetsanopoulos, A. N. (2010). Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications (The Springer International Series in Engineering and Computer Science, 209) (Softcover reprint of hardcover 1st ed. 1993). Springer.

C. Lampiran

1. Link GitHub:
   https://github.com/berlianm/Case-Based01-ML

2. Link Hasil Pengerjaan Pemrograman:
   https://colab.research.google.com/drive/1YGmEim8v-gHseW7Bdrem6OSq4ch9JiJi?usp=sharing

3. Link Dokumen Laporan:
   https://docs.google.com/document/d/1Myrv-tdpTvMU_tqnE5jcQWbI5_DmqoYB2uPkvSQxb40/edit?usp=sharing

4. Link Dokumen Presentasi:
   https://docs.google.com/presentation/d/1-F2ADSCEbR3ZrJY65EgGTvLxF_DeGwRpw2Wrv4d-6YQ/edit?usp=sharing

5. Link Video Presentasi:
   https://drive.google.com/file/d/1xHQGWyGgdEOP-67KIEyNg3BZTbRziz-m/view?usp=sharing

6. Link Google Drive:
   https://drive.google.com/drive/folders/1V1fbY2YWxbgsTd3T9afNhe17T7ILB2YN