

Case-Based 1
Pembelajaran Mesin CII3C3
Semester Ganjil 2022/2023



Disusun Oleh :
1301204142 - Jovidia Laviosa

Kode Dosen : BDP
IF-44-01

PROGRAM STUDI INFORMATIKA
FAKULTAS INFORMATIKA

LEMBAR PERNYATAAN

Saya yang bertanda tangan dibawah ini mengerjakan tugas **Case-Based 1 Pembelajaran Mesin CII3C3** dengan jujur tanpa meminta dan menerima bantuan dari siapapun. Tidak meniru ataupun melanggar aturan perkuliahan dan kode etik akademisi. Karena Saya menyadari bahwa Tuhan Yang Maha Kuasa Maha Melihat perbuatan saya, dan saya menyadari bahwa nilai hasil kecurangan tidak akan membawa keberkahan dalam perkuliahan, pekerjaan, dan hidup saya seterusnya. Saya menyadari bahwa pengetahuan yang saya peroleh lebih penting dari pada hanya sekedar nilai hasil kecurangan. Saya Menyadari kecurangan tidak membawa manfaat sedikitpun bagi diri saya dan sebaliknya akan merusak harkat, martabat, kepercayaan diri, dan mental saya sendiri.

Bandung, 4 November 2022



Jovidia Laviosa
1301204142

KATA PENGANTAR

Puji dan syukur marilah kita panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan nikmat sehat sehingga kami dapat menyelesaikan Laporan Tugas **Case-Based 1 Pembelajaran Mesin CII3C3 Semester Ganjil 2022/2023**.

Terima kasih saya ucapkan kepada Bapak Bedy Purnama yang telah membantu melalui pemberian materi di kelas. Saya menyadari, bahwa laporan Tugas Case Based 1 yang saya buat ini masih jauh dari kata sempurna dalam penulisannya. Oleh karena itu, saya sangat mengharapkan kritik dan saran yang membangun dari semua pembaca, agar penulis menjadi lebih baik lagi di masa mendatang.

Dan semoga laporan Tugas Case Based 1 ini bisa menambah wawasan para pembaca dan dapat bermanfaat untuk perkembangan dan peningkatan ilmu pengetahuan.

Bandung, 4 November 2022

Penulis

BAB 1

PENDAHULUAN

1.1 Abstraksi

Aritmia adalah gangguan kesehatan yang membuat pengidapnya mengalami detak jantung tidak teratur, baik lebih cepat maupun lebih lambat. Permasalahan irama jantung ini pada umumnya tidak berbahaya. Namun, jika detak jantung sudah mulai terasa tidak biasa, maka bisa berakibat fatal hingga menyebabkan kematian mendadak.


Basis data ini berisi 279 atribut, 206 di antaranya bernilai linier dan sisanya nominal. Mengenai studi H. Altay Guvenir: "Tujuannya adalah untuk membedakan antara ada dan tidak adanya aritmia jantung dan untuk mengklasifikasikannya dalam salah satu dari 16 kelompok. Kelas 01 mengacu pada EKG 'normal' kelas 02 hingga 15 mengacu pada kelas yang berbeda aritmia dan kelas 16 mengacu pada sisa yang tidak terklasifikasi. Untuk itu dibuatlah deep learning machine agar dapat memprediksi dan mengklasifikasikannya dengan tepat menggunakan metode CNN (Convolutional Neural Network).

Database: Arrhythmia

Class code :	Class :	Number of instances:
01	Normal	245
02	Ischemic changes (Coronary Artery Disease)	44
03	Old Anterior Myocardial Infarction	15
04	Old Inferior Myocardial Infarction	15
05	Sinus tachycardia	13
06	Sinus bradycardia	25
07	Ventricular Premature Contraction (PVC)	3
08	Supraventricular Premature Contraction	2
09	Left bundle branch block	9
10	Right bundle branch block	50
11	1. degree AtrioVentricular block	0
12	2. degree AV block	0
13	3. degree AV block	0
14	Left ventricular hypertrophy	4
15	Atrial Fibrillation or Flutter	5
16	Others	22


1.2 Ikhtisar Kumpulan Data Arrhythmia

1. Pertama yang harus dilakukan adalah mengimport library untuk dapat membaca dataset

```
0 d  # Mengimpor library untuk membaca dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

- Library pandas berfungsi untuk membuat tabel, mengubah dimensi data dan mengecek data.
- Library numpy memberi kemudahan dalam komputasi khususnya tipe data numerik.
- Library matplotlib.pyplot berfungsi untuk membuat sebuah plot
- Library seaborn untuk membentuk grafik

2. Mengimport data dari link yang sudah disediakan di soal arrhythmia.data (NIM genap)

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

	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 x 280 columns

3. Menampilkan detail informasi dari data menggunakan df.describe()

↳ Eksplorasi Data

1d #Menampilkan detail informasi terkait data
df.describe()

	0	1	2	3	4	5	6	7	8	9	..
count	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	..
mean	46.471239	0.550885	166.188053	68.170354	88.920354	155.152655	367.207965	169.949115	90.004425	33.676991	..
std	16.466631	0.497955	37.170340	16.590803	15.364394	44.842283	33.385421	35.633072	25.826643	45.431434	..
min	0.000000	0.000000	105.000000	6.000000	55.000000	0.000000	232.000000	108.000000	0.000000	-172.000000	..
25%	36.000000	0.000000	160.000000	59.000000	80.000000	142.000000	350.000000	148.000000	79.000000	3.750000	..
50%	47.000000	1.000000	164.000000	68.000000	86.000000	157.000000	367.000000	162.000000	91.000000	40.000000	..
75%	58.000000	1.000000	170.000000	79.000000	94.000000	175.000000	384.000000	179.000000	102.000000	66.000000	..
max	83.000000	1.000000	780.000000	176.000000	188.000000	524.000000	509.000000	381.000000	205.000000	169.000000	..

8 rows x 275 columns

4. Membuat keterangan kolom sesuai dengan data yang diberikan di arrhythmia.names

1d #Menambahkan keterangan sesuai data
column_indices = [0,1,2,3,4,5,6,7,8,9,10,11,12,15,16,17,18,166,164,161,163, 167, 279]
new_names = [
 "Age", "Sex", "Height", "Weight",
 "QRS_duration", "P-R_interval",
 "Q-T_interval", "T_interval",
 "P_interval", "QRS", "T", "P",
 "QRST", "Heart_rate", "DI_Q_wave_width", "DI_R_wave_width", "DI_S_wave_width",
 "DI_P_wave_amplitude", "DI_R_wave_amplitude", "DI_Q_wave_amplitude", "DI_S_wave_amplitude",
 "DI_T_wave_amplitude", "Arrhythmia"
]
old_names = df.columns[column_indices]
df.rename(columns=dict(zip(old_names, new_names)), inplace=True)
df.head(5)

	Age	Sex	Height	Weight	QRS_duration	P-R_interval	Q-T_interval	T_interval	P_interval	QRS	...	270	271	272	273	274	275	276	277	278	Arrhythmia
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 x 280 columns

5. Menampilkan data missing value. Hasilnya tidak ada data yang missing value

0d #Menampilkan data missing value
df.isna().sum()

Age	0
Sex	0
Height	0
Weight	0
QRS_duration	0
...	..
275	0
276	0
277	0
278	0
Arrhythmia	0

Length: 280, dtype: int64

6. Membuat kerangka data untuk diagram garis amplitudo per millisecond

```
✓ 1d #Membuat kerangka data untuk diagram garis amplitudo
wf = pd.DataFrame(columns=['Milliseconds', 'Amplitude'])

arr = df.copy()

hr = arr.loc[0, 'Heart_rate']
hb_dur_min = 1.0 / hr
hb_dur_sec = hb_dur_min * 60
hb_dur_millisec = hb_dur_sec * 1000

p_start = arr.loc[0, 'P_interval']
p_amplitude = arr.loc[0, 'DI_P_wave_amplitude']

pr_interval = arr.loc[0, 'P-R_interval']

qrs_interval = arr.loc[0, 'QRS_duration']

qt_interval = arr.loc[0, 'Q-T_interval']

q_interval = arr.loc[0, 'DI_Q_wave_width']
q_amplitude = arr.loc[0, 'DI_Q_wave_amplitude']

r_interval = arr.loc[0, 'DI_R_wave_width']
r_amplitude = arr.loc[0, 'DI_R_wave_amplitude']

s_interval = arr.loc[0, 'DI_S_wave_width']
s_amplitude = arr.loc[0, 'DI_S_wave_amplitude']

t_interval = arr.loc[0, 'T_interval']
t_amplitude = arr.loc[0, 'DI_T_wave_amplitude']

print(p_start, p_amplitude)

wf = wf.append(pd.DataFrame({'Milliseconds':0, 'Amplitude':0}, index=[len(wf)]))
wf = wf.append(pd.DataFrame({'Milliseconds':p_start/2, 'Amplitude':[p_amplitude]}, index=[len(wf)]))
wf = wf.append(pd.DataFrame({'Milliseconds':p_start, 'Amplitude':0}, index=[len(wf)]))

wf = wf.append(pd.DataFrame({'Milliseconds':pr_interval, 'Amplitude':0}, index=[len(wf)]))

q_interval += pr_interval
wf = wf.append(pd.DataFrame({'Milliseconds':q_interval, 'Amplitude':q_amplitude}, index=[len(wf)]))

r_interval += q_interval
wf = wf.append(pd.DataFrame({'Milliseconds':r_interval, 'Amplitude':r_amplitude}, index=[len(wf)]))

s_interval += r_interval
wf = wf.append(pd.DataFrame({'Milliseconds':s_interval, 'Amplitude':s_amplitude}, index=[len(wf)]))

s_end = pr_interval+qrs_interval
if s_end < s_interval:
    s_end = s_interval+10
wf = wf.append(pd.DataFrame({'Milliseconds':s_end, 'Amplitude':0}, index=[len(wf)]))

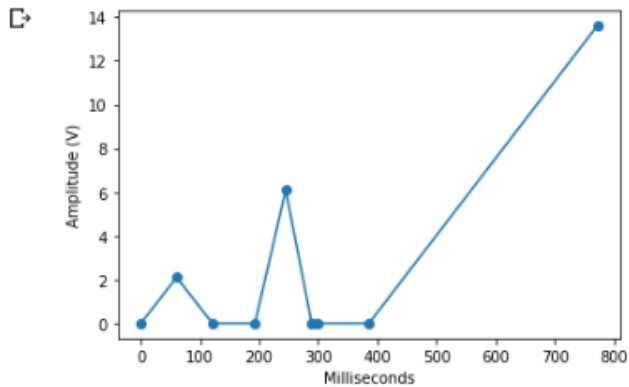
# estimate st_segment as half the time remaining
t_interval = s_end+t_interval
st_segment = s_end + ((t_interval - s_end) / 2)
wf = wf.append(pd.DataFrame({'Milliseconds':st_segment, 'Amplitude':0}, index=[len(wf)]))

#wf = wf.append(pd.DataFrame({'Seconds':p_interval, 'Amplitude':[p_amplitude]}, index=[len(wf)]))
t_interval = s_end+t_interval
wf = wf.append(pd.DataFrame({'Milliseconds':t_interval, 'Amplitude':t_amplitude}, index=[len(wf)]))
```

Hasilnya :

✓
0 d

```
#Diagram Amplitudo  
plt.plot(wf['Milliseconds'], wf['Amplitude'], '-o')  
plt.xlabel("Milliseconds")  
plt.ylabel("Amplitude (V)")  
plt.show()
```



7. Membuat diagram batang dari deskripsi data yang diberikan

✓
0 d

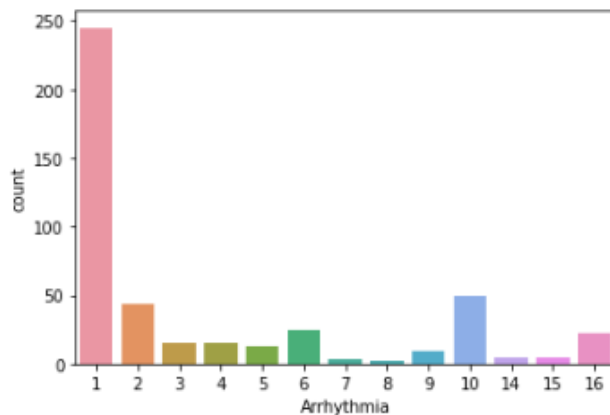
```
[212] #List arrhythmia - dari deskripsi data yang diberikan  
#Class code:      Class:                                     Number of instances:  
#01               Normal                                     245  
#02               Ischemic changes (Coronary Artery Disease) 44  
#03               Old Anterior Myocardial Infarction          15  
#04               Old Inferior Myocardial Infarction          15  
#05               Sinus tachycardy                            13  
#06               Sinus bradycardy                            25  
#07               Ventricular Premature Contraction (PVC)     3  
#08               Supraventricular Premature Contraction      2  
#09               Left bundle branch block                    9  
#10               Right bundle branch block                   50  
#11               1. degree AtrioVentricular block            0  
#12               2. degree AV block                          0  
#13               3. degree AV block                          0  
#14               Left ventricle hypertrophy                   4  
#15               Atrial Fibrillation or Flutter              5  
#16               Others                                      22
```

Hasilnya :

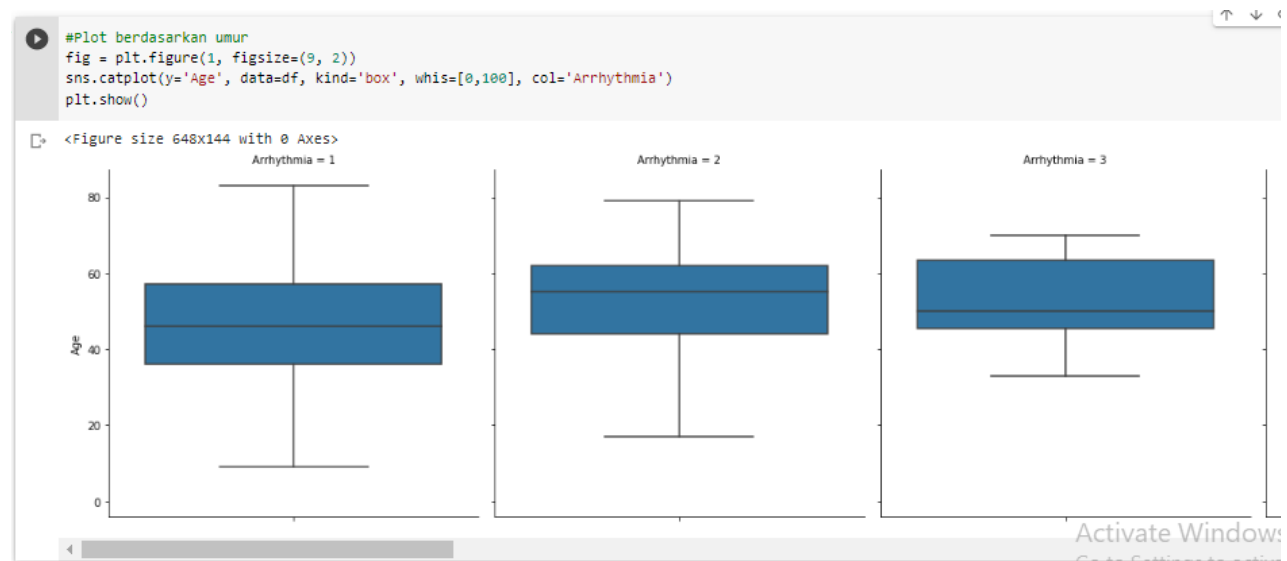
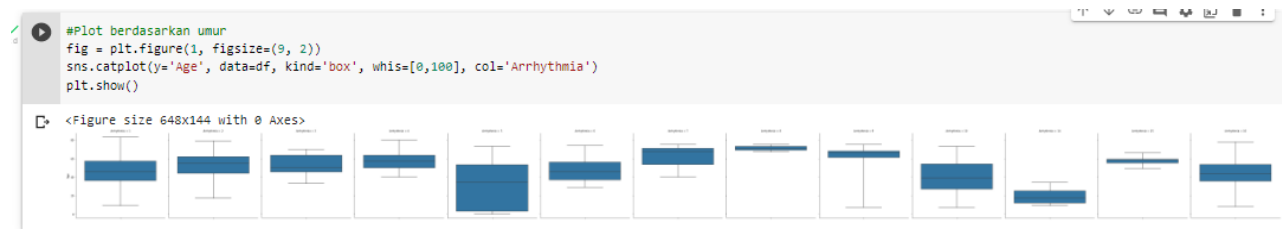
✓
0 d

```
#Diagram plot dari arrhythmia  
sns.countplot(x='Arrhythmia', data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0edd779bd0>



8. BoxPlot berdasarkan umur



BAB 2

PRA-PEMROSESAN DATA

Menghilangkan/drop kolom yang merupakan data ganjil atau dirasa kurang sesuai.

1. Menghilangkan kolom yang berkorelasi dalam data

▼ Pra-pemrosesan Data

```
# Temukan dan hilangkan kolom yang berkorelasi dalam data
def correlated(dataset, threshold=0.8):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i,j])>threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return list(col_corr)

cor_cols = correlated(df,0.8)

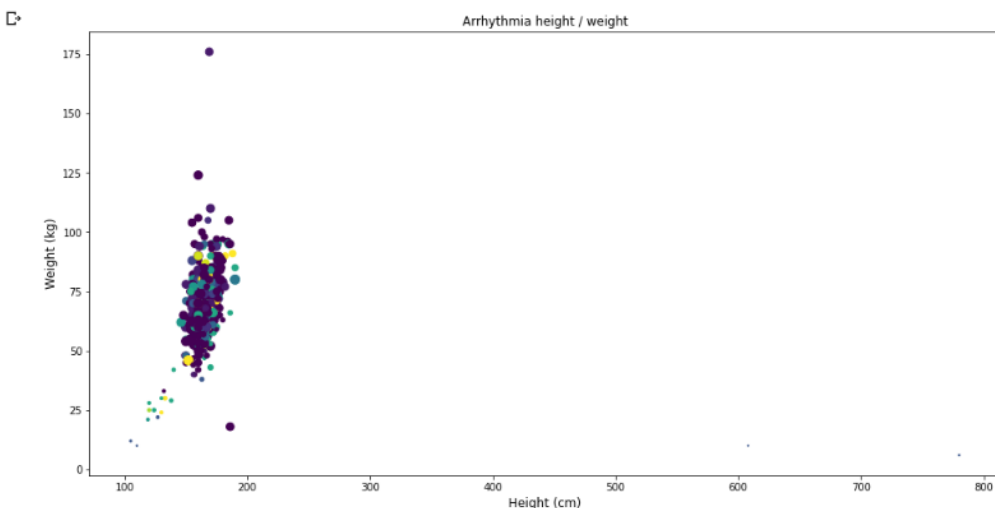
df = df.drop(labels=cor_cols,axis=1)
df
```

	Age	Sex	Height	Weight	QRS_duration	P-R_interval	Q-T_interval	T_interval	P_interval	QRS	...	256	257	258	259	261	262	264	267	274
0	75	0	190	80	91	193	371	174	121	-16	...	5.1	17.7	70.7	-0.4	13.5	-4.0	0.0	25.5	0.0
1	56	1	165	64	81	174	401	149	39	25	...	2.6	11.8	34.6	-0.4	11.0	-2.4	0.0	21.6	0.0
2	54	0	172	95	138	163	386	185	102	96	...	2.2	-3.0	20.7	1.3	11.1	-3.4	0.0	11.5	0.0
3	55	0	175	94	100	202	380	179	143	28	...	3.3	28.8	63.1	0.1	15.2	-3.7	0.0	36.8	0.0
4	75	0	190	80	88	181	360	177	103	-16	...	4.9	16.2	63.2	-0.2	9.1	-0.9	0.0	21.7	0.0
...

2. Membuat anomali tinggi dan berat badan

```
# Anomali tinggi dan berat badan
def plotArrhythmia():
    plt.figure(figsize=(16,8))
    plt.title('Arrhythmia height / weight')
    plt.scatter(x=df['Height'], y=df['Weight'], c=df['Arrhythmia'], s=(1+df['Age']))
    plt.xlabel('Height (cm)', fontsize=12)
    plt.ylabel('Weight (kg)', fontsize=12)
    #plt.legend(arrhythmia['Arrhythmia'])
    plt.show()

plotArrhythmia()
```



Ternyata dari data scatter plot yang ditampilkan, terdapat beberapa data yang masih kurang sesuai atau dirasa salah. Seperti tinggi badan dan berat badan yang abnormal. Oleh karena itu, diperlukan menghapus kolom dari data yang salah agar hasil perhitungan pengklasifikasian mendapatkan hasil yang akurat.

3. Menampilkan kolom dengan tinggi badan lebih dari 600 cm

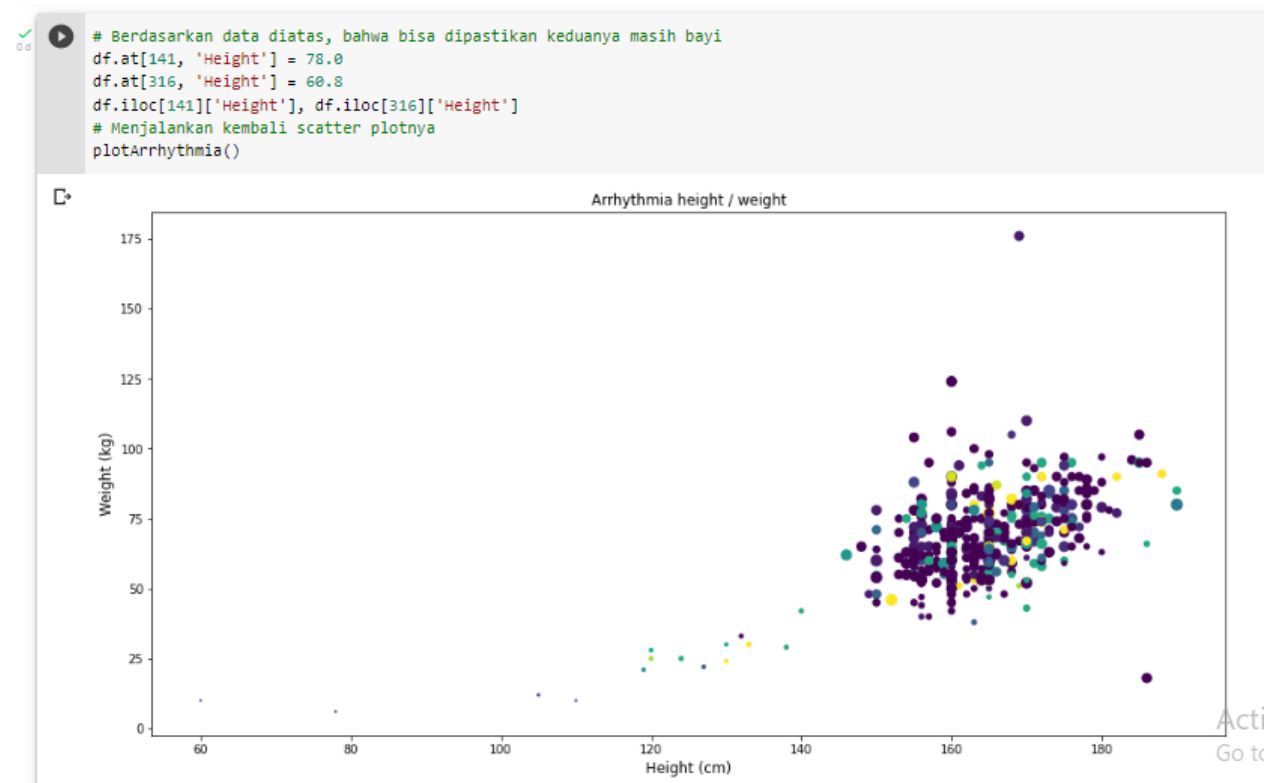
```
# Menampilkan kolom yang mempunyai data tinggi badan lebih dari 600
df.loc[df['Height'] > 600]
```

	Age	Sex	Height	Weight	QRS_duration	P-R_interval	Q-T_interval	T_interval	P_interval	QRS	...	256	257	258	259	261	262	264	267	274	Arrhythmia
141	1	1	780	6	85	165	237	150	106	88	...	1.7	21.9	32.4	0.1	17.2	-7.6	0.0	13.5	0.0	5
316	0	0	608	10	83	126	232	128	60	125	...	4.8	-14.1	11.8	2.8	8.3	-9.3	0.0	-11.8	0.0	5

2 rows x 220 columns

Terdapat 2 orang dengan tinggi badan abnormal yaitu 780 dan 608 cm.

Ternyata dari 2 orang tersebut berusia 1 atau dibawah satu dapat dipastikan keduanya masih bayi.



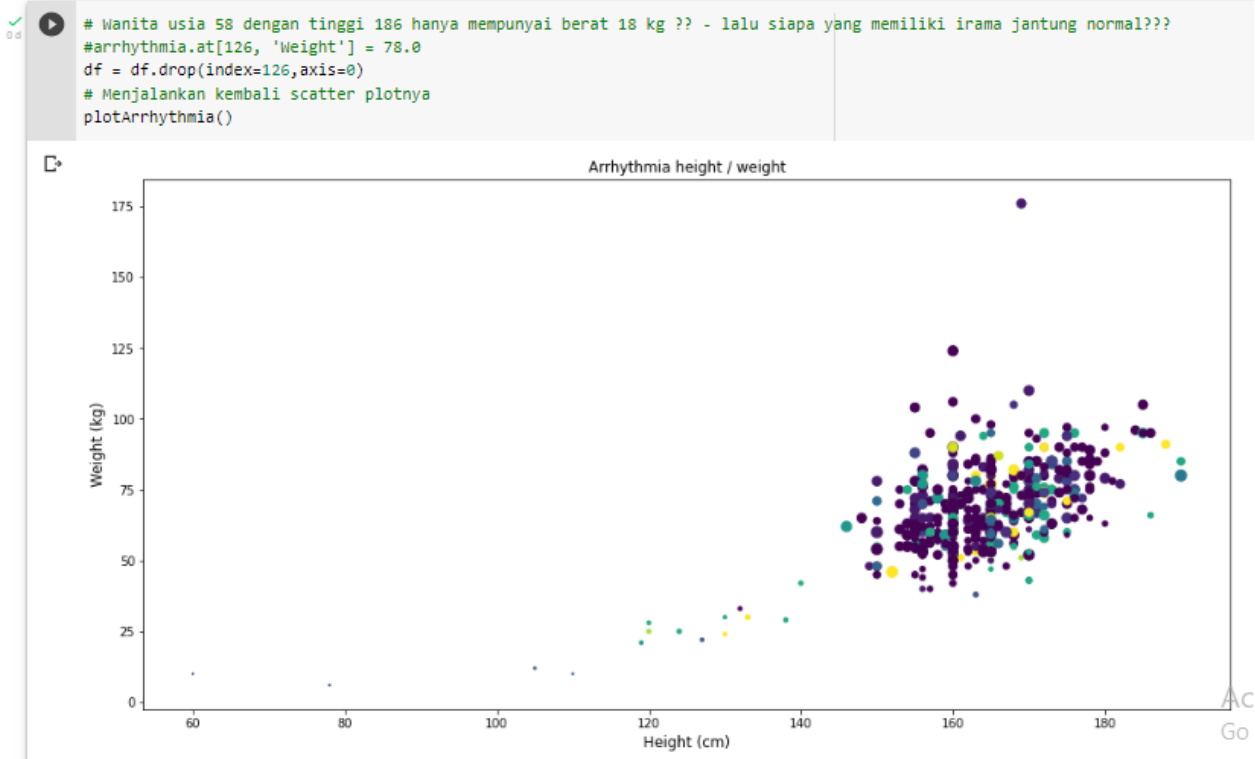
4. Menampilkan kolom dengan berat badan dibawah 20 kg.

```
# Menampilkan kolom yang mempunyai berat dibawah 20 kg
df.loc[df['Weight'] < 20]
```

	Age	Sex	Height	Weight	QRS_duration	P-R_interval	Q-T_interval	T_interval	P_interval	QRS	...	256	257	258	259	261	262	264	267	274	Arrhythmia
60	1	0	110	10	80	121	287	156	67	126	...	2.6	2.4	20.6	0.6	8.0	-6.9	0.0	-1.7	0.0	5
126	58	0	186	18	87	166	372	150	96	-1	...	2.9	0.2	23.9	0.0	7.6	-5.4	0.0	2.6	0.0	1
141	1	1	78	6	85	165	237	150	106	88	...	1.7	21.9	32.4	0.1	17.2	-7.6	0.0	13.5	0.0	5
316	0	0	60	10	83	126	232	128	60	125	...	4.8	-14.1	11.8	2.8	8.3	-9.3	0.0	-11.8	0.0	5
320	3	0	105	12	69	155	240	133	64	93	...	0.4	0.3	1.6	-0.2	4.8	-4.3	0.0	3.3	0.0	5

5 rows x 220 columns

Diantaranya terdapat wanita berusia 58 tahun dengan tinggi 186 yang hanya mempunyai berat badan 18 kg dan memiliki irama jantung yang normal, tentu saja itu tidak mungkin terjadi. Sehingga kita harus menghapusnya dari data.



5. Menunjukan individu yang paling berat. Dalam hal ini lebih dari 170 kg.

```
[223] # Menunjukan individu yang paling berat - Menampilkan wanita berusia 53 - dengan penyakit arteri koroner - apakah salah pengetikan ?
df.loc[df['Weight'] > 170]
```

	Age	Sex	Height	Weight	QRS_duration	P-R_interval	Q-T_interval	T_interval	P_interval	QRS	...	256	257	258	259	261	262	264	267	274	Arrhythmia
213	53	0	169	176	111	166	339	200	100	86	...	2.8	10.6	35.2	-0.8	14.3	-1.6	0.0	29.7	0.0	2

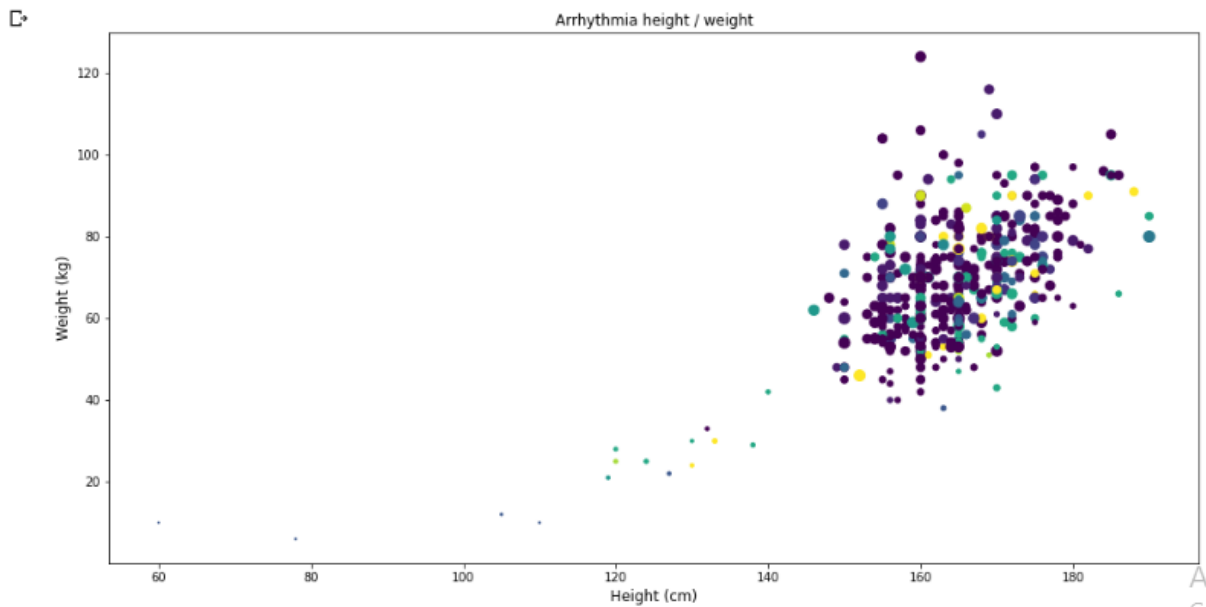
1 rows x 220 columns

Menampilkan wanita berusia 53 tahun dengan penyakit arteri koroner. Kita asumsikan saja salah penulisan. Dapat dilihat dari persebaran data bahwa data sudah benar.

```

# mari kita asumsikan bahwa itu adalah kesalahan transkripsi dari catatan tulisan tangan
df.at[213, 'Weight'] = 116.0
#Menjalankan kembali scatter plotnya
plotArrhythmia()

```



```

df['Arrhythmia'].value_counts()

```

```

1      244
10     50
2      44
6      25
16     22
3      15
4      15
5      13
9       9
15      5
14      4
7       3
8       2
Name: Arrhythmia, dtype: int64

```

BAB 3

METODE ALGORITMA CNN

1. Mengimpor Library

```
import numpy as np
import pandas as pd
import keras
import tensorflow as tf
import ssl
import math
import matplotlib.pyplot as plt
import operator

from collections import defaultdict
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, KFold
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import normalize, StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import LinearSVC, SVC
from sklearn.metrics import confusion_matrix, classification_report, f1_score, accuracy_score
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, LSTM
from tensorflow.keras.layers import Conv1D, MaxPooling1D, BatchNormalization, GlobalMaxPooling1D, MaxPooling1D
from keras.regularizers import l2, l1
from keras.utils import to_categorical
```

2. Membuat train dan test

```
# Pisahkan atribut Data dan Kelas
df_data = df.iloc[:, :-1]
df_class = df.iloc[:, -1]

df_data = df_data.replace('?', np.NaN)

# Hapus kolom yang tidak diinginkan
# Menghapus atribut yang memiliki lebih dari 40% nilai yang hilang.
thresh = len(df_data) * 0.4
df_data.dropna(thresh = thresh, axis = 1, inplace = True)

imp_mean = SimpleImputer(missing_values=np.NaN, strategy='median')
imputer = imp_mean.fit(df_data)
df_imp = imputer.transform(df_data)
df_data = pd.DataFrame(df_imp)

# Attribute Scaling
# Menormalkan nilai kecuali untuk label kelas untuk setiap atribut menggunakan StandardScaler.
std_scaler = StandardScaler()
x_scaled = std_scaler.fit_transform(df_data.values)
df_data = pd.DataFrame(x_scaled, index = df_data.index)

print(df_data.shape)
```

(451, 218)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got FutureWarning,

```

#Membagi menjadi data pelatihan dan pengujian
X_train, X_test, Y_train, Y_test = train_test_split(df_data, df_class, test_size=0.3, shuffle = True, stratify = df_class, random_state=43)

# Membagi menjadi data pelatihan dan validasi
#X_train, X_val, Y_train, Y_val = train_test_split(X_trainval, Y_trainval, test_size=0.2, shuffle = True, stratify = Y_trainval, random_state=43)

print(X_train.shape, Y_train.shape, X_test.shape, Y_test.shape)

```

(315, 218) (315,) (136, 218) (136,)

CNN

```

# Membagi menjadi data pelatihan dan pengujian
X_train_cnn, X_val_cnn, Y_train_cnn, Y_val_cnn = train_test_split(X_train, Y_train, test_size=0.2, random_state=0)

Xtrain = np.expand_dims(X_train, 2)
Ytrain = to_categorical(Y_train)
Xval = np.expand_dims(X_test, 2)
Yval = to_categorical(Y_test)
print(Xtrain.shape, Ytrain.shape, Xval.shape, Yval.shape)

print(np.bincount(Y_train))
print(np.bincount(Y_test))
print(np.bincount(Y_train_cnn))
print(np.bincount(Y_val_cnn))

```

(315, 218, 1) (315, 17) (136, 218, 1) (136, 17)

```

[ 0 170 31 11 11 9 17 2 1 6 35 0 0 0 3 4 15]
[ 0 74 13 4 4 4 8 1 1 3 15 0 0 0 1 1 7]
[ 0 131 25 8 11 7 14 2 1 5 29 0 0 0 3 4 12]
[ 0 39 6 3 0 2 3 0 0 1 6 0 0 0 0 0 3]

```

3. Menentukan bobot kelas untuk dataset yang tidak seimbang

```

from sklearn.utils.class_weight import compute_class_weight
class_weights = compute_class_weight(class_weight = "balanced", classes= np.unique(Y_train), y= Y_train)
class_weights_dict = dict(zip([1,2,3,4,5,6,7,8,9,10,14,15,16], class_weights))
class_weights_dict[0] = 0
class_weights_dict[11] = 0
class_weights_dict[12] = 0
class_weights_dict[13] = 0
print(class_weights.sum())
print(class_weights_dict)

```

66.27433619511419

```

{1: 0.1425339366515837, 2: 0.7816377171215881, 3: 2.202797202797203, 4: 2.202797202797203, 5: 2.692307692307692!

```

4. Mendefinisikan model CNN 1D yang sudah seimbang

```

#model
model = Sequential()

model.add(Conv1D(filters=64, kernel_size=10,activation='relu',kernel_initializer='he_uniform', input_shape=(278,1)))
model.add(Conv1D(filters=128, kernel_size=10,activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(17, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

```

Model: "sequential_21"

Layer (type)	Output Shape	Param #
conv1d_26 (Conv1D)	(None, 269, 64)	704
conv1d_27 (Conv1D)	(None, 260, 128)	82048
dropout_13 (Dropout)	(None, 260, 128)	0
flatten_13 (Flatten)	(None, 33280)	0
dense_63 (Dense)	(None, 128)	4259968
dense_64 (Dense)	(None, 64)	8256
dense_65 (Dense)	(None, 17)	1105

Total params: 4,352,081
 Trainable params: 4,352,081
 Non-trainable params: 0

5. Latih model dan simpan pemeriksaan akurasi validasi terbaik

```

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
batch_size= 16
no_epochs = 20

earlystop = EarlyStopping(monitor='val_accuracy', patience=20)
checkpoint = ModelCheckpoint('model-epoch-{epoch:03d}-valacc-{val_accuracy:03f}.h5', verbose=1, monitor='val_accuracy', save_best_only=True, mode='auto')

# Generate the fit model
hist = model.fit(xtrain, ytrain,
                epochs=no_epochs,
                batch_size=batch_size,
                validation_data=(xval, yval),
                callbacks=[earlystop, checkpoint],
                class_weight = class_weights_dict)

```

```

Epoch 1/20
20/20 [=====] - ETA: 0s - loss: 4.0992 - accuracy: 0.1994
Epoch 0001: val_accuracy improved from -inf to 0.51471, saving model to model-epoch-001-valacc-0.514706.h5
20/20 [=====] - 2s 103ms/step - loss: 4.0992 - accuracy: 0.1994 - val_loss: 2.6113 - val_accuracy: 0.5147
Epoch 2/20
20/20 [=====] - ETA: 0s - loss: 2.0121 - accuracy: 0.6139
Epoch 0002: val_accuracy did not improve from 0.51471
20/20 [=====] - 2s 94ms/step - loss: 2.0121 - accuracy: 0.6139 - val_loss: 2.0967 - val_accuracy: 0.2794
Epoch 3/20
20/20 [=====] - ETA: 0s - loss: 0.9706 - accuracy: 0.5538
Epoch 0003: val_accuracy did not improve from 0.51471
20/20 [=====] - 2s 92ms/step - loss: 0.9706 - accuracy: 0.5538 - val_loss: 2.1444 - val_accuracy: 0.4118
Epoch 4/20
20/20 [=====] - ETA: 0s - loss: 0.5303 - accuracy: 0.7500
Epoch 0004: val_accuracy did not improve from 0.51471
20/20 [=====] - 2s 92ms/step - loss: 0.5303 - accuracy: 0.7500 - val_loss: 2.2720 - val_accuracy: 0.4559
Epoch 5/20
20/20 [=====] - ETA: 0s - loss: 0.3105 - accuracy: 0.7658
Epoch 0005: val_accuracy did not improve from 0.51471
20/20 [=====] - 2s 92ms/step - loss: 0.3105 - accuracy: 0.7658 - val_loss: 2.1369 - val_accuracy: 0.4412
Epoch 6/20
20/20 [=====] - ETA: 0s - loss: 0.1557 - accuracy: 0.8291
Epoch 0006: val_accuracy improved from 0.51471 to 0.58088, saving model to model-epoch-006-valacc-0.580882.h5
20/20 [=====] - 2s 96ms/step - loss: 0.1557 - accuracy: 0.8291 - val_loss: 2.1868 - val_accuracy: 0.5809
Epoch 7/20
20/20 [=====] - ETA: 0s - loss: 0.0674 - accuracy: 0.9589
Epoch 0007: val_accuracy did not improve from 0.58088
20/20 [=====] - 2s 94ms/step - loss: 0.0674 - accuracy: 0.9589 - val_loss: 2.2295 - val_accuracy: 0.5588
Epoch 8/20
20/20 [=====] - ETA: 0s - loss: 0.0450 - accuracy: 0.9209
Epoch 0008: val_accuracy improved from 0.58088 to 0.58088, saving model to model-epoch-008-valacc-0.580882.h5

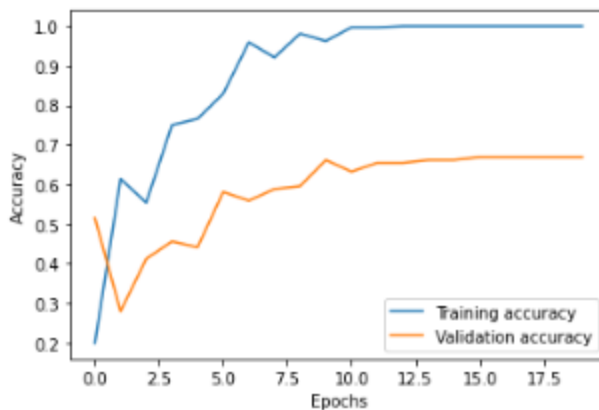
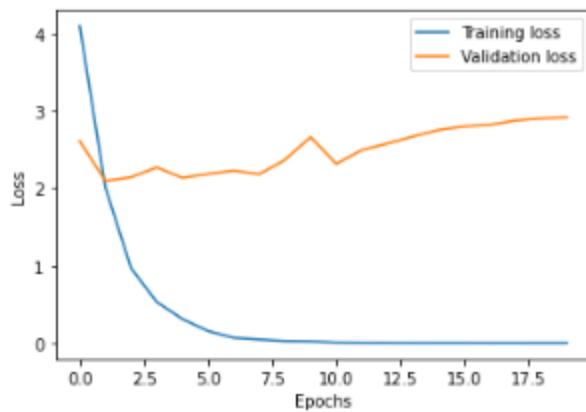
```


6. Memvisualisasikan akurasi pelatihan dan validasi

```
[417] sub=0
# visualizing losses and accuracy
train_loss = hist.history['loss'][sub:]
val_loss = hist.history['val_loss'][sub:]
train_accu = hist.history['accuracy'][sub:]
val_accu = hist.history['val_accuracy'][sub:]
xc = range(no_epochs)

fig1 = plt.figure()
fig1.patch.set_facecolor('white')
plt.plot(xc, train_loss, label='Training loss')
plt.plot(xc, val_loss, label='Validation loss')
plt.legend(loc="upper right")
plt.xlabel('Epochs')
plt.ylabel('Loss')
fig1.show()

fig2 = plt.figure()
fig2.patch.set_facecolor('white')
plt.plot(xc, train_accu, label='Training accuracy')
plt.plot(xc, val_accu, label='Validation accuracy')
plt.legend(loc="lower right")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
fig2.show()
```



BAB 4

EVALUASI

Dari hasil percobaan diatas dapat disimpulkan bahwa dengan menggunakan metode algoritma CNN (Convolutional Neural Network), akurasi dari training dan validasi sudah lumayan cukup walaupun belum sempurna. Lalu ada perbedaan di training loss dan validation loss dimana masih perlu dilakukan percobaan dan analisa lebih lanjut terkait pengambilan data yang sesuai. Keakuratan dari hasil percobaan ini adalah 0.6985 atau 70%

```
print('Accuracy of Weighted 1D-CNN model - ',round(accuracy_score(list(Y_test),Ypred),4))
```

Accuracy of Weighted 1D-CNN model - 0.6985

BAB 5

PRESENTASI VIDEO

Link Slide :

<https://drive.google.com/file/d/1X0r88syVNMaWpjYCW4iznwPgF3Ndl3hd/view?usp=sharing>

Link Presentasi : https://youtu.be/IwnPWIchf_0