Laporan Case-Based 1
Mata Kuliah Pembelajaran Mesin
Artificial Neural Network (ANN)

Muhamad Azmi Rizkifar – 1301218586 – IFX-45-GAB – IKN

"Saya mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi"

Program Studi Sarjana Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2022

BAGIAN I
Ikhtisar Data Yang Dipilih

Dataset yang akan dipilih dalam pengimplementasian *ANN* adalah dataset *Arrhythmia* yang didapatkan dari *UCI Machine Learning Repository*. Tujuan penulis adalah untuk membangun sebuah model klasifikasi yang dapat memprediksi antara ada dan tidak adanya aritmia jantung dan mengklasifikasikannya dalam salah satu dari 16 kelompok data. Data yang didapat disajikan dalam bentuk tabulasi yang telah di konversi ke file Excel (xlsx).

75	0	190	80	91	193	371	174	121	-16	13	64	-2	?		63	0	52	44	0	0	32
56	1	165	64	81	174	401	149	39	25	37	-17	31	?		53	0	48	0	0	0	24
54	0	172	95	138	163	386	185	102	96	34	70	66		23	75	0	40	80	0	0	24
55	0	175	94	100	202	380	179	143	28	11	-5	20	?		71	0	72	20	0	0	48
75	0	190	80	88	181	360	177	103	-16	13	61	3	?	?		0	48	40	0	0	28
13	0	169	51	100	167	321	174	91	107	66	52	88	?		84	0	36	48	0	0	20
40	1	160	52	77	129	377	133	77	77	49	75	65	?		70	0	44	0	0	0	24
49	1	162	54	78	0	376	157	70	67	7	8	51	?		67	0	44	36	0	0	24
44	0	168	56	84	118	354	160	63	61	69	78	66		84	64	0	40	0	0	0	20
50	1	167	67	89	130	383	156	73	85	34	70	71	?		63	0	44	40	0	0	28
62	0	170	72	102	135	401	156	83	72	71	68	72	?		70	20	36	48	0	0	36
45	1	165	86	77	143	373	150	65	12	37	49	26	?		72	0	40	28	0	0	20
54	1	172	58	78	155	382	163	81	-24	42	41	-13	?		73	0	72	0	0	0	24
30	0	170	73	91	180	355	157	104	68	51	60	63	?		56	0	92	0	0	0	32
44	1	160	88	77	158	399	163	94	46	20	45	40	?		72	0	80	0	0	0	28
47	1	150	48	75	132	350	169	65	36	45	68	40	?		76	0	48	0	0	0	24
47	0	171	59	82	145	347	169	61	77	75	77	75	?		67	0	48	0	0	0	20
46	1	158	58	70	120	353	122	52	57	49	-2	54	?		70	0	48	0	0	0	24
73	0	165	63	91	154	392	175	83	73	-24	61	42	?		66	0	44	56	0	0	20
57	1	166	72	82	181	399	158	79	-12	28	50	1	?		66	0	56	16	0	0	28
28	1	160	58	83	251	383	189	183	50	39	46	43	?		76	16	36	28	0	0	32
45	0	169	67	90	122	336	177	78	81	78	67	80	?		66	0	36	24	0	0	20
36	1	153	75	71	132	364	169	82	62	56	45	60	?		77	0	44	12	0	0	24

Gambar 1 Dataset Arrhythmia Data

Dari keseluruhan data yang ada, terdapat total 452 data pasien yang dilengkapi dengan 279 kolom atau nilai fitur dan 1 kolom target yaitu kategori aritmia jantung. Pada kolom target, Terdapat 16 jenis kelompok data aritmia jantung yang menunjukkan apakah seorang pasien termasuk ke dalam salah satu dari beberapa jenis kelompok tersebut. Data jenis kelompok tersebut berisi :

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

Dataset Arrhythmia memiliki missing values pada beberapa fiturnya sehingga perlu dilakukannya pra-pemrosesan data dengan menghapus beberapa fitur atau dengan mengisi nilai rata-rata dari suatu fitur dengan bantuan Python. Selain itu juga terdapat beberapa fitur yang

seluruh isinya bernilai 0 dan juga terdapat beberapa fitur yang hampir seluruh isinya bernilai 0. Dikarenakan pada dataset tidak terdapat label pada setiap kolomnya, maka penulis memberikan label untuk 20 kolom pertama, dan sisa kolom lainnya diberikan nama dengan prefix COL[nomor urut fitur].

	Age	Sex	Height	Weight	QRS_Duration	P_R_Interval	Q_T_Interval	T_Interval	P_Interval	QRS	...	COL271	COL272	COL273	COL274	COL275	COL276
0	75.0	0.0	190.0	80.0	91.0	193.0	371.0	174.0	121.0	-16.0	...	0.0	9.0	-0.9	0.0	0.0	0.9
1	56.0	1.0	165.0	64.0	81.0	174.0	401.0	149.0	39.0	25.0	...	0.0	8.5	0.0	0.0	0.0	0.2
2	54.0	0.0	172.0	95.0	138.0	163.0	386.0	185.0	102.0	96.0	...	0.0	9.5	-2.4	0.0	0.0	0.3
3	55.0	0.0	175.0	94.0	100.0	202.0	380.0	179.0	143.0	28.0	...	0.0	12.2	-2.2	0.0	0.0	0.4
4	75.0	0.0	190.0	80.0	88.0	181.0	360.0	177.0	103.0	-16.0	...	0.0	13.1	-3.6	0.0	0.0	-0.1

Gambar 2 Head Data Arrhythmia

Adapun beberapa fitur yang seluruh isinya bernilai nol yang ditampilkan pada Gambar 3 berikut :

```
     S'_Wave  COL68  COL70  COL84  COL132  COL133  COL140  COL142  COL144  \
0        0.0    0.0    0.0    0.0     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     0.0
2        0.0    0.0    0.0    0.0     0.0     0.0     0.0     0.0     0.0
3        0.0    0.0    0.0    0.0     0.0     0.0     0.0     0.0     0.0
4        0.0    0.0    0.0    0.0     0.0     0.0     0.0     0.0     0.0
..       ...    ...    ...    ...     ...     ...     ...     ...     ...
447      0.0    0.0    0.0    0.0     0.0     0.0     0.0     0.0     0.0
448      0.0    0.0    0.0    0.0     0.0     0.0     0.0     0.0     0.0
449      0.0    0.0    0.0    0.0     0.0     0.0     0.0     0.0     0.0
450      0.0    0.0    0.0    0.0     0.0     0.0     0.0     0.0     0.0
451      0.0    0.0    0.0    0.0     0.0     0.0     0.0     0.0     0.0

     COL146  COL152  COL157  COL158  COL165  COL205  COL265  COL275
0       0.0     0.0     0.0     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
2       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
3       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
4       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
..      ...     ...     ...     ...     ...     ...     ...     ...
447     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
448     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
449     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
450     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
451     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

[452 rows x 17 columns]
```

Gambar 3 Daftar Fitur Yang Semua Isinya Bernilai 0

BAGIAN II
Pra-Pemrosesan Data

Sebelum dilakukannya pembuatan model dan prediksi data, perlu dilakukannya tahap pra-pemrosesan data dengan harapan untuk mendapatkan hasil prediksi yang lebih akurat. Penulis menemukan bahwa pada data tersebut terdapat beberapa *missing values* pada fiturnya dan perlu untuk dilakukan proses eliminasi dan mengisi nilai rata-rata pada beberapa fitur yang memiliki nilai missing values dengan jumlah yang tidak terlalu banyak. Selain itu juga terdapat beberapa fitur yang semua isinya bernilai 0, dan beberapa fitur lainnya memiliki banyak nilai nol. Berikut merupakan penjabaran tahapan pra-pemrosesan yang dilakukan oleh penulis :

1. **Eliminasi fitur dan replace missing values**

   Terdapat dua cara untuk penanganan *missing values* yang penulis lakukan, yaitu dengan cara menghapus fitur yang memiliki banyak *missing values* dan mengisi nilai *missing values* dengan nilai rata-rata dari fitur tersebut apabila memiliki jumlah *missing values* yang tidak terlalu banyak. Dikarenakan missing value pada dataset ini didefinisikan dengan karakter "?" yang mengakibatkan library pandas tidak bisa mendeteksi *null value*, maka dilakukan mekanisme pengubahan nilai *Non-Number* ke dalam nilai *NaN (Not a Number)*.

```
# ubah value non number ke NaN
for i in data.columns:
    cnt=0
    for row in data[i]:
        try:
            float(row)
            pass
        except ValueError:
            data.loc[cnt, i] = np.nan
        cnt+=1
```

   Dari proses pengubahan tersebut, didapatkan hasil sebanyak 408 *missing values* dan selanjutnya dilakukan pencarian lokasi *missing values* tersebut yang berada pada beberapa fitur yang ditampilkan pada gambar dibawah.

```
# cek kembali missing value pada data (yang terbaca NaN)
print(data.isnull().values.any())

# tampilkan jumlah missing value dalam dataset
print(data.isnull().sum().sum())
```

```
True
408
```

```
# tampilkan daftar missing value dari setiap kolom
show_missing_values()
```

```
T => 8 missing values
P => 22 missing values
QRST => 1 missing values
J => 376 missing values
Heart_Rate => 1 missing values
```

Setelah didapatkan beberapa fitur dengan jumlah missing valuesnya, dilakukan penghapusan pada fitur dengan label **T**, **P**, dan **J** karena memiliki banyak missing values. Kemudian dilakukan pengisian missing values pada sisa fitur lainnya.

```
[ ]  # drop data pada kolom 'T', 'P', dan 'J'
     delete = ['T', 'P', 'J']
     data.drop(delete, axis=1, inplace=True)
```

```
[ ]  # tampilkan kembali daftar missing value setelah penghapusan data
     show_missing_values()

     QRST => 1 missing values
     Heart_Rate => 1 missing values
```

```
[ ]  # replace missing value pada kolom 'QRST' dan 'Heart_Rate' dengan nilai rata-rata
     for i in data.columns[data.isnull().any()].tolist():
       data[str(i)].fillna((data[str(i)].mean()), inplace=True)
```

```
[ ]  # tampilkan kembali daftar missing value setelah penghapusan data
     show_missing_values()

     # cek kembali missing value pada data (yang terbaca NaN)
     print(data.isnull().values.any())

     # tampilkan jumlah missing value dalam dataset
     print(data.isnull().sum().sum())

     Sudah tidak terdapat missing values
     False
     0
```

2. **Eliminasi fitur yang seluruh valuesnya bernilai 0**

Pada tahapan ini, penulis melakukan pencarian data terlebih dahulu dengan ketentuan semua fitur yang seluruh valuesnya bernilai 0.

```
# Get data yang semua valuenya berisi 0
all_zero_values = data.loc[:, (data == 0).all()]

print(all_zero_values)
```

```
[452 rows x 17 columns]
```

Dari hasil pencarian tersebut, ditemukan sebanyak 17 fitur yang selanjutnya akan dilakukan penghapusan/eliminasi.

```
[ ]  # drop semua fitur yang semua valuenya = 0
     delete_all_zero_val = all_zero_values.columns.tolist()
     data.drop(delete_all_zero_val, axis=1, inplace=True)

     # tampilkan kembali data yang semua valuenya berisi 0
     print(data.loc[:, (data == 0).all()])

     Empty DataFrame
     Columns: []
```

3. **Eliminasi fitur dengan rasio jumlah nilai 0 lebih dari 50%**

Pada tahapan ini, penulis juga melakukan pencarian data terlebih dahulu dengan ketentuan semua fitur yang memiliki values bernilai 0.

```
[ ]  # Get data yang memiliki value berisi 0
     zero_values = data.loc[:, (data == 0).any()]

     print('Jumlah fitur yang memiliki value nol (0) => {} fitur'.format(len(zero_values.columns)), '\n')
     print(zero_values)
```

```
Jumlah fitur yang memiliki value nol (0) => 231 fitur

      Age  Sex  P_R_Interval  P_Interval   QRS  QRST  Q_Wave  R_Wave  S_Wave  \
0    75.0  0.0         193.0       121.0 -16.0  -2.0     0.0    52.0    44.0
1    56.0  1.0         174.0        39.0  25.0  31.0     0.0    48.0     0.0
2    54.0  0.0         163.0       102.0  96.0  66.0     0.0    40.0    80.0
3    55.0  0.0         202.0       143.0  28.0  20.0     0.0    72.0    20.0
4    75.0  0.0         181.0       103.0 -16.0   3.0     0.0    48.0    40.0
..    ...  ...           ...         ...   ...   ...     ...     ...     ...
447  53.0  1.0         199.0       117.0 -37.0 -27.0     0.0    52.0    24.0
448  37.0  0.0         137.0        73.0  86.0  79.0     0.0    44.0    36.0
449  36.0  0.0         176.0       116.0 -85.0 -70.0    16.0    40.0    40.0
450  32.0  1.0         106.0        63.0  54.0  43.0     0.0    56.0     0.0
451  78.0  1.0         127.0        78.0  28.0  47.0     0.0    44.0    28.0
```

Dari hasil pencarian tersebut, ditemukan sebanyak 231 fitur dimana masing-masing fiturnya memiliki isi yang bernilai 0. Untuk mengetahui jumlah nilai nol dari masing-masing fitur tersebut, dilakukan kembali pencarian jumlah nilai nolnya pada masing-masing fitur seperti gambar dibawah :

```
[ ]  delete_zero_value = []

     for i in zero_values:
       # Tampilkan jumlah value nol (0) dari setiap fitur
       print('{} => {} zero values'.format(str(i), (data[str(i)] == 0).sum()))

       # zero value > 50% jumlah data
       if (data[str(i)] == 0).sum() > (len(zero_values) * 0.5):
         delete_zero_value.append(i)
```

```
Age => 1 zero values
Sex => 203 zero values
P_R_Interval => 18 zero values
P_Interval => 12 zero values
QRS => 3 zero values
QRST => 6 zero values
Q_Wave => 334 zero values
R_Wave => 3 zero values
S_Wave => 192 zero values
R'_Wave => 448 zero values
COL21 => 3 zero values
COL22 => 451 zero values
COL23 => 447 zero values
COL24 => 447 zero values
COL25 => 450 zero values
COL26 => 450 zero values
COL27 => 448 zero values
COL28 => 347 zero values
```

Penulis melampirkan screenshot yang hanya sebagian hasilnya saja, karena dirasa terlalu panjang dikarenakan terdapat 231 baris output yang ditampilkan dari hasil pencarian tersebut. Setelah dilakukan pencarian data, penulis melakukan penghapusan fitur dengan ketentuan memiliki jumlah nilai 0 lebih dari 50% jumlah nilai pada setiap fiturnya.

```
[ ]  print('Jumlah fitur dengan zero value > 50% = {} dari {} fitur'.format(
         len(delete_zero_value),
         len(zero_values.columns)
     ))

     # drop fitur yang rasio jumlah nilai 0 nya > 50%
     data.drop(delete_zero_value, axis=1, inplace=True)
```

```
Jumlah fitur dengan zero value > 50% = 131 dari 231 fitur
```

Ditemukan sebanyak 131 dari 231 fitur yang memiliki jumlah nilai 0 yang lebih dari 50% jumlah nilai pada setiap fiturnya, dan kemudian dilakukan eliminasi sebanyak 131 fitur. Dari hasil pra-pemrosesan pada beberapa tahapan diatas, didapatkan sisa sebanyak 129 dari 280 fitur yang telah dilakukan pra-pemrosesan.

```
[ ]  # summary pre-process
     print('Jumlah semua fitur yang telah di Pre-Process = {} dari {} fitur'.format(
         len(data.columns),
         features_count_before
     ))

     Jumlah semua fitur yang telah di Pre-Process = 129 dari 280 fitur
```

4. Normalisasi data dan feature selection

Pada tahapan akhir pra-pemrosesan, penulis melakukan normalisasi data yang bertujuan untuk memastikan record pada dataset tetap konsisten dan menghilangkan redundansi data agar data memiliki kualitas yang lebih baik. Normalisas dibutuhkan karena beberapa atribut dari dataset ini memiliki skala atau rentang yang berbeda-beda sehingga terdapat ketimpangan dimana terdapat data yang terlampau tinggi dan terdapat data yang terlampau rendah.

Penulis menggunakan bantuan library dari sklearn untuk melakukan normalisasi dan feature selection. Pertama, penulis menggunakan function *SimpleImputer()* untuk penanganan nilai data yang kosong/hilang apabila masih terdapat data kosong yang belum tertangani pada tahapan sebelumnya. Kemudian penulis melakukan normalisasi *Min-Max* dengan mengubah sekumpulan data menjadi skalai mulai dari 0 (Min) hingga 1 (Max) dengan bantuan function *MinMaxScaler()*. Berikut terlampir formula perhitungan normalisasi Min-Max dan penerapannya :

$$X_{norm} = \frac{x' - min(x)}{max(x) - min(x)} (new_{max}(x) - new_{min}(x)) + new_{min}(x)$$

```
[ ]  # Melakukan normalisasi data
     x_df = data.loc[:, data.columns != 'Target']
     x = np.array(x_df)

     imputer = SimpleImputer(missing_values=np.nan, strategy='median')
     imputer = imputer.fit(x)
     x_imputer = imputer.transform(x)

     df_feature = data.drop('Target' , axis=1)
     df_selected_target = data[['Target']]

     scaler = MinMaxScaler()
     x_norm = scaler.fit_transform(x_imputer)
```

Selanjutnya, penulis melakukan *feature selection* dengan tujuan untuk mendapatkan fitur-fitur optimal yang memiliki korelasi tinggi dan mengeliminasi fitur-fitur yang tidak berpengaruh atau fitur yang tidak relevan. Penulis menggunakan metode feature selection *Chi Square* dan menggunakan bantuan function SelectKBest() dengan memilih 80 fitur terbaik yang akan digunakan pada proses prediksi dataset nanti.

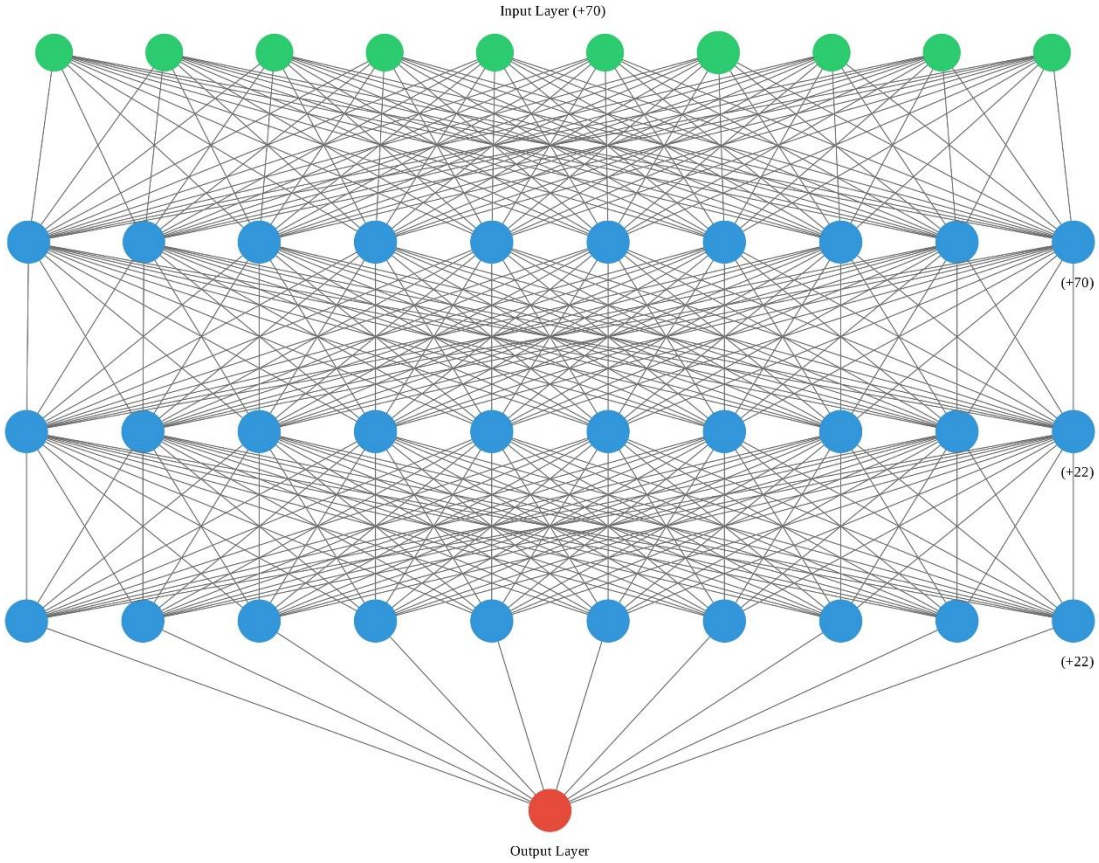
```
[ ]  # features selection dengan chi square
     chi2_selector = SelectKBest(chi2, k=80)
     chi2_selector.fit(x_norm, df_selected_target)

     cols = chi2_selector.get_support(indices=True)
     df_selected_features = df_feature.iloc[:, cols]

     ohe = OneHotEncoder()
     df_selected_target = ohe.fit_transform(df_selected_target).toarray()

[ ]  df_selected_features.head()
```

	Age	Sex	QRS_Duration	Q_T_Interval	T_Interval	P_Interval	Heart_Rate	R_Wave	S_Wave	COL21	...	COL252	COL257	COL258	COL262	COL267	COL269	COL272	COL277	CO
0	75.0	0.0	91.0	371.0	174.0	121.0	63.000000	52.0	44.0	32.0	...	15.2	5.1	17.7	13.5	3.9	62.9	9.0	2.9	
1	56.0	1.0	81.0	401.0	149.0	39.0	53.000000	48.0	0.0	24.0	...	9.5	2.6	11.8	11.0	2.6	43.4	8.5	2.1	
2	54.0	0.0	138.0	386.0	185.0	102.0	75.000000	40.0	80.0	24.0	...	10.0	2.2	-3.0	11.1	3.4	48.2	9.5	3.4	
3	55.0	0.0	100.0	380.0	179.0	143.0	71.000000	72.0	20.0	48.0	...	15.0	3.3	28.8	15.2	3.0	68.0	12.2	2.6	
4	75.0	0.0	88.0	360.0	177.0	103.0	74.463415	48.0	40.0	28.0	...	15.2	4.9	16.2	9.1	2.9	48.9	13.1	3.9	

5 rows × 80 columns

BAGIAN III
Penerapan Algoritma

Pada proses penerapan algoritmanya, penulis menggunakan algoritma *Artifical Neural Network* (*ANN*) dikarenakan algoritma ini merupakan algoritma yang sesuai dengan dataset yang didapatkan untuk mengklasifikasikan ada atau tidaknya aritmia jantung pada pasien berdasarkan 16 kelompok data. Algoritma *ANN* ini dibangun dengan 1 input layer sebanyak 80 nodes, 2 hidden layer masin-masing sebanyak 32 nodes, dan 1 output layer sebanyak 1 nodes.



Berikut penjabaran implementasi ANN dalam penyelesaian studi kasus Arrhythmia Data dengan bantuan library *Tensorflow* pada *Python* :

1. **Pembuatan sample data latih dan data test**

   Pada penerapannya, dilakukan pemisahan dataset untuk data latih dan data test dengan rasio 80:20 dengan detail 80% untuk data latih dan 20% untuk data test.

```
[64] # Membuat sampel data latih dan test
     x_train, x_test, y_train, y_test = train_test_split(df_selected_features,
                                                         df_selected_target,
                                                         test_size=0.2,
                                                         random_state=24,
                                                         shuffle=True)

     print('Train set:', x_train.shape, y_train.shape)
     print('Test set:', x_test.shape, y_test.shape)

     Train set: (361, 80) (361, 13)
     Test set: (91, 80) (91, 13)
```

2. Pembuatan model arsitektur

Model arsitektur yang dibangun menggunakan 1 input layer yang terdiri dari 80 nodes dan 80 input shape yang merepresentasikan 80 fitur yang digunakan, 2 hidden layer dengan masing-masing 32 nodes, dan 1 output layer dengan 1 nodes. Fungsi aktivasi yang digunakan pada input layer dan hidden layer adalah relu.

```
[67] # Membuat model arsitektur
     model = tf.keras.models.Sequential()
     model.add(tf.keras.layers.Dense(units=80, input_shape=(80,), activation='relu')) # input layer ada 80 nodes menyesuikan dengan b
     model.add(tf.keras.layers.Dense(units=32, activation='relu')) # menggunakan 2 hidden layer, dengan masing-masing jumlah node seb
     model.add(tf.keras.layers.Dense(units=32, activation='relu'))
     model.add(tf.keras.layers.Dense(units=1)) # output layer

     model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 80)                6480

 dense_5 (Dense)             (None, 32)                2592

 dense_6 (Dense)             (None, 32)                1056

 dense_7 (Dense)             (None, 1)                 33

=================================================================
Total params: 10,161
Trainable params: 10,161
Non-trainable params: 0
_________________________________________________________________
```

3. Proses compile model

Pada proses compile modelnya, penulis menggunakan *Mean Absolute Error* (*MAE*) sebagai paramter loss dengan menggunakan *Adam* sebagai optimizernya dengan learning rate sebesar 0,001. Untuk metriknya digunakan *accuracy* sebagai acuan akurasi pada saat proses training modelnya.

```
# Compile model
model.compile(loss='mean_absolute_error',
              optimizer=tf.keras.optimizers.Adam(0.001),
              metrics=['accuracy'])
```

4. Training Model

Pada proses training modelnya, penulis mendefinisikan paramter x_train sebagai fitur yang akan di latih, y_train sebagai target fitur, epoch sebanyak 50 iterasi, data validation yang digunakan untuk memvalidasi model dan mencegah *overfitting*, dan callbacks untuk menghentikan iterasi apabila nilai akurasi dan akurasi validasinya lebih besar dari 90%.

```
# Melakukan fitting model
fitted_model = model.fit(x_train, y_train, epochs=50, validation_data=(x_test, y_test), callbacks=callbacks)
```

```
Epoch 13/50
12/12 [==============================] - 0s 5ms/step - loss: 0.3612 - accuracy: 0.7942 - val_loss: 0.3954 - val_accuracy: 0.9045
Epoch 14/50
12/12 [==============================] - 0s 6ms/step - loss: 0.3204 - accuracy: 0.8668 - val_loss: 0.2976 - val_accuracy: 0.8580
Epoch 15/50
12/12 [==============================] - 0s 4ms/step - loss: 0.2507 - accuracy: 0.9043 - val_loss: 0.2692 - val_accuracy: 0.8859
Epoch 16/50
12/12 [==============================] - 0s 5ms/step - loss: 0.2421 - accuracy: 0.9043 - val_loss: 0.2670 - val_accuracy: 0.8859
Epoch 17/50
 1/12 [=>............................] - ETA: 0s - loss: 0.2446 - accuracy: 0.9231
 Desired accuracy and validation_accuracy > 90%
12/12 [==============================] - 0s 7ms/step - loss: 0.2186 - accuracy: 0.9160 - val_loss: 0.2525 - val_accuracy: 0.9045
```

Link Google Collab : https://colab.research.google.com/drive/1mOPZ9qemNPlMcmRo4GOSzXBykN-gE7dW?usp=sharing

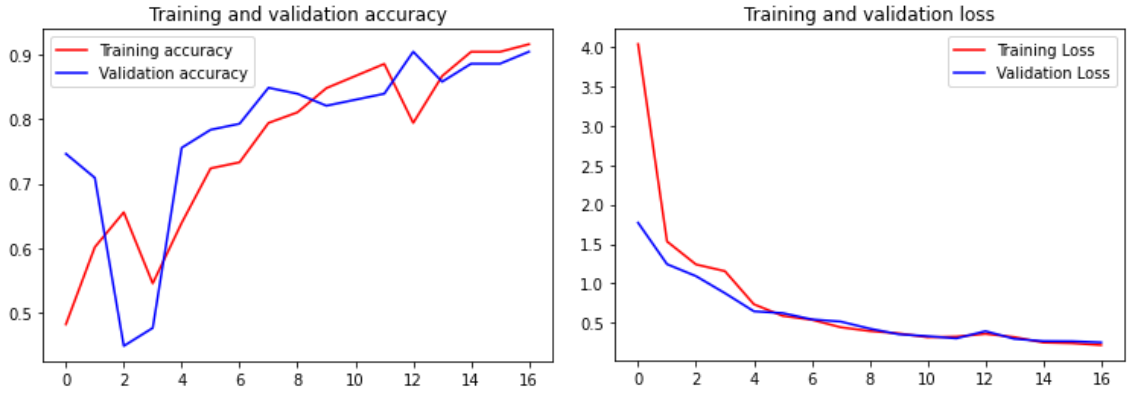
Link Github : https://github.com/azmirizkifar20/Machine-Learning---Implementasi-ANN

BAGIAN IV
Evaluasi Hasil

Pengukuran performansi model dilihat dari nilai akurasi yang didapatkan dari epoch ke 17 dengan nilai akurasi sebesar 0,9160 dan nilai akurasi validasi sebesar 0,9045.

```
Epoch 16/50
12/12 [==============================] - 0s 5ms/step - loss: 0.2421 - accuracy: 0.9043 - val_loss: 0.2670 - val_accuracy: 0.8859
Epoch 17/50
 1/12 [=>............................] - ETA: 0s - loss: 0.2446 - accuracy: 0.9231
 Desired accuracy and validation_accuracy > 90%
12/12 [==============================] - 0s 7ms/step - loss: 0.2186 - accuracy: 0.9160 - val_loss: 0.2525 - val_accuracy: 0.9045
```

Dari hasil akurasi tersebut, penulis memvisualisasikan hasilnya kedalam bentuk grafik yang dapat dilihat pada gambar berikut :



Pada tahap evaluasi model dengan menggunakan data test didapatkan nilai loss sebesar 0,25 dan nilai akurasinya sebesar 0,904.

```
[ ]  # evaluasi model
     score = model.evaluate(x_test, y_test, verbose=0)

     print('Test loss:', score[0])
     print('Test accuracy:', score[1])

     Test loss: 0.2517230212688446
     Test accuracy: 0.9044801592826843
```

Dari hasil nilai evaluasi model berdasarkan nilai akurasi dan validasi akurasinya, didapatkan nilai yang diharapkan yaitu nilai akurasi yang lebih dari 90%. Hal ini membuktikan bahwa model bekerja dengan baik dalam mempelajari data-data latih tersebut ketika memvalidasi input dan target pada datanya. hasil akurasi tersebut menunjukkan bahwa model yang dibangun memiliki performansi yang baik untuk digunakan dalam memprediksi data baru. Hasil nilai akurasi dari data training dan data test tersebut tidak terpaut jauh yang dapat menunjukkan bahwa model tidak underfit ataupun overfit.