

LAPORAN
CASE-BASED 1
MATA KULIAH PEMBELAJARAN MESIN
MEMBANGUN MODEL MENGGUNAKAN ALGORITMA NEURAL NETWORK

Nama Dosen : Agus Hartoyo, S.T.
Kode Dosen : AHY



Disusun Oleh :

Aisyah Dliya Ramadhanti

1301201154

IF4406

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2022/2023

BAB 1

DATASET

1.1 Pegumpulan Data

Pada tugas case based 1 Mata Kuliah Pembelajaran Mesin, kami diminta untuk menganalisis algoritma supervised learning (ANN/MLP/RNN/LSTM/CNN) dari kumpulan data yang tersedia. Laporan kali ini menganalisis dataset Arrhythmia (untuk NIM genap) yang nantinya akan diprediksi menggunakan algoritma ANN. Berikut adalah dataframe dari data yang akan digunakan.

```
data = pd.read_csv('arrhythmia_csv.csv', sep = ",")
data
```

	age	sex	height	weight	QRSduration	PRinterval	QTinterval	QTinterval	QTinterval	QRS	...	chV6_QuaveAmp	chV6_RuaveAmp	chV6_SuaveAmp	chV6_RPuaveAmp	chV6_SPuaveAmp	chV6_PuaveAmp
0	75	0	190	80	91	193	371	174	121	-16	...	0.0	9.0	-0.9	0.0	0.0	0.0
1	56	1	165	64	81	174	401	149	39	25	...	0.0	8.5	0.0	0.0	0.0	0.0
2	54	0	172	95	138	163	386	185	102	96	...	0.0	9.5	-2.4	0.0	0.0	0.0
3	55	0	175	94	100	202	380	179	143	28	...	0.0	12.2	-2.2	0.0	0.0	0.0
4	75	0	190	80	88	181	360	177	103	-16	...	0.0	13.1	-3.6	0.0	0.0	0.0
...
447	53	1	160	70	80	199	382	154	117	-37	...	0.0	4.3	-5.0	0.0	0.0	0.0
448	37	0	190	85	100	137	361	201	73	86	...	0.0	15.6	-1.6	0.0	0.0	0.0
449	36	0	166	68	108	176	365	194	116	-85	...	0.0	16.3	-28.6	0.0	0.0	0.0
450	32	1	155	55	93	106	386	218	63	54	...	-0.4	12.0	-0.7	0.0	0.0	0.0
451	78	1	160	70	79	127	364	138	78	28	...	0.0	10.4	-1.8	0.0	0.0	0.0

452 rows x 280 columns

Dataset arrhythmia adalah data yang membedakan antara ada atau tidak adanya arrhythmia jantung yang kemudian diklasifikasikan ke salah satu dari 16 kategori. Karakteristik dari dataset tersebut yaitu multivariate. Sedangkan karakteristik atribut pada dataset tersebut yaitu kategorikal, integer dan real.

```
[4] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 452 entries, 0 to 451
Columns: 280 entries, age to class
dtypes: float64(125), int64(155)
memory usage: 988.9 KB
```

Berdasarkan informasi di atas, diperoleh informasi bahwa dataset arrhythmia terdiri dari 452 baris dan 280 kolom/atribut dengan tipe data integer dan float.

1.1.1 Pembagian Data

Dikarenakan jumlah atribut yang terlalu banyak, maka diperlukan pembagian data agar atribut yang digunakan lebih sedikit sehingga data yang kita gunakan untuk bangun model menjadi lebih sederhana. Hal ini berguna agar memudahkan kita dalam pemilihan feature.

```
data.drop(data.columns[8:-3],axis=1, inplace=True)  
data.shape
```

```
(452, 11)
```

Sesuai gambar di atas, atribut data ke-9 dihapus hingga atribut data ke-4 dari terakhir sehingga hanya 11 atribut yang akan diolah lebih lanjut.

1.1.2 Deskripsi Data

Berikut deskriptif statistik data Arrythmia dengan 11 atribut.

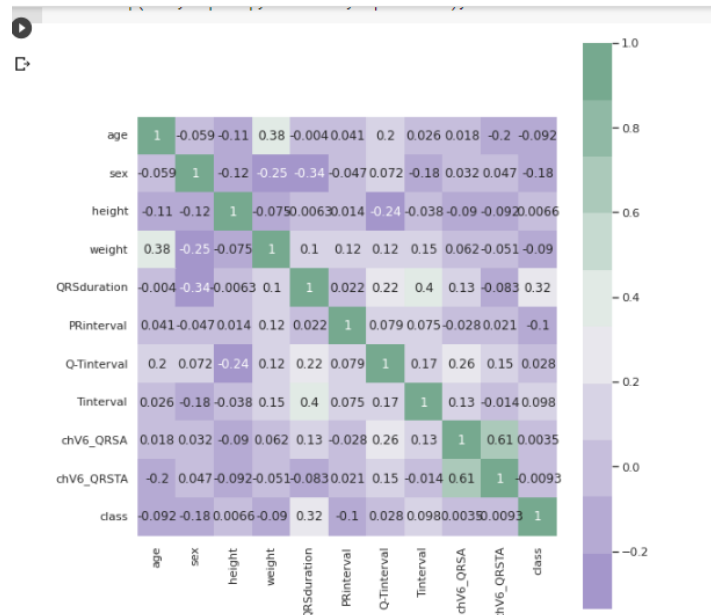
```
[7] data.describe()
```

	age	sex	height	weight	QRSduration	PRinterval	Q-Tinterval	Tinterval	chV6_QRSA	chV6_QRSTA	class
count	452.000000	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	19.326106	29.473230	3.880531
std	16.466631	0.497955	37.170340	16.590803	15.364394	44.842283	33.385421	35.633072	13.503922	18.493927	4.407097
min	0.000000	0.000000	105.000000	6.000000	55.000000	0.000000	232.000000	108.000000	-44.200000	-38.600000	1.000000
25%	36.000000	0.000000	160.000000	59.000000	80.000000	142.000000	350.000000	148.000000	11.450000	17.550000	1.000000
50%	47.000000	1.000000	164.000000	68.000000	86.000000	157.000000	367.000000	162.000000	18.100000	27.900000	1.000000
75%	58.000000	1.000000	170.000000	79.000000	94.000000	175.000000	384.000000	179.000000	25.825000	41.125000	6.000000
max	83.000000	1.000000	780.000000	176.000000	188.000000	524.000000	509.000000	381.000000	88.800000	115.900000	16.000000

1.2 Visualisasi Dataset

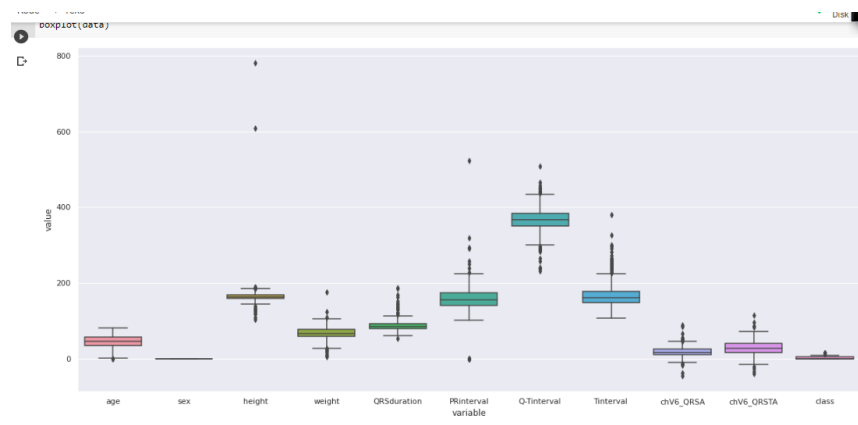
1.2.1 Heatmap Korelasi

Berikut adalah heatmap korelasi dari 11 atribut pada dataset arrhythmia.



1.2.2 Boxplot

Pada data terdapat beberapa outlier pada beberapa atribut yang ada. Outlier ini dapat dilihat pada boxplot di bawah ini.



Outliers yang ada pada data menyebabkan data menjadi tidak bersih. Oleh karena itu, outliers yang ada pada data harus dihilangkan dengan cara menormalisasi data pada bagian data cleaning nanti.

1.3 Kualitas Dataset

Dataset yang kita miliki saat ini masih terdapat noise (data masih kotor). Hal ini dikarenakan data yang kita miliki mengandung banyak missing value dan juga outliers pada data. Data akan dibersihkan pada tahap data cleaning.

1.3.1 Missing Value

Missing value yang terdapat pada data, menyebabkan data yang kita punya menjadi kotor. Oleh karena itu, kita perlu menghilangkan missing value agar data yang kita punya menjadi bersih. Handling missing value dapat menggunakan teknik menghapus kolom/atribut data atau dengan metode imputation. Pada data awal (data dengan 280 atribut) terdapat 408 missing value. Hal tersebut dapat dilihat pada gambar di bawah ini.

```
data.isnull().sum().sum()
408
```

Setelah data disederhanakan menjadi 11 atribut, missing value pada data tersebut menjadi 0. Hal ini dapat dilihat pada gambar di bawah ini.

```
[29] data.isnull().sum().sum()
0
```

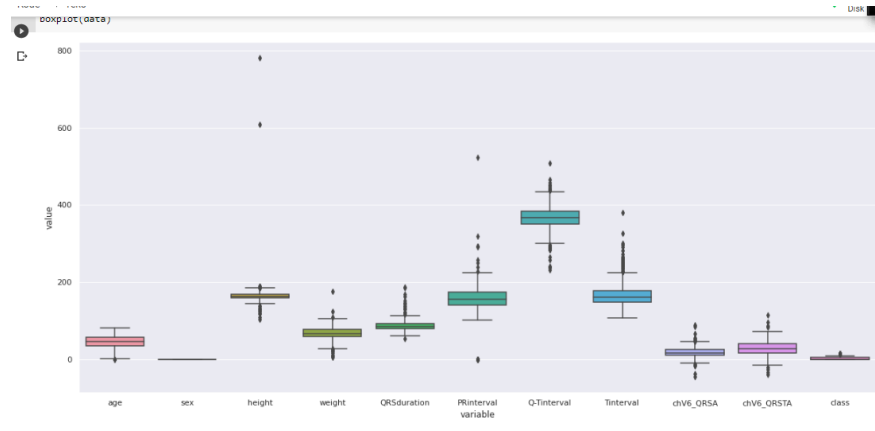
```
[30] #melihat persen missing value dari setiap kolom data
total = data.isnull().sum().sort_values(ascending=False)
percent = (data.isnull().sum()/data.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

	Total	Percent
age	0	0.0
sex	0	0.0
height	0	0.0
weight	0	0.0
QRSduration	0	0.0
PRinterval	0	0.0
Q-Tinterval	0	0.0
Tinterval	0	0.0
chV6_QRSA	0	0.0
chV6_QRSTA	0	0.0
class	0	0.0

Dikarenakan pada data sekarang tidak ada missing value, maka kita tidak perlu lagi melakukan metode handling missing value.

1.3.2 Outliers pada Data

Data yang kita miliki masih memiliki banyak outliers pada beberapa atribut. Hal tersebut dapat dilihat pada boxplot berikut.




Outliers menyebabkan distribusi data menjadi tidak normal. Selain itu, outliers dapat menimbulkan.

BAB 2

PRE-PROCESSING DATA

Pada data yang kita punya saat ini, tidak terdapat missing value maupun inkonsistensi data sehingga kita tidak perlu menerapkan metode handling missing value. Akan tetapi, pada data tersebut terdapat beberapa outliers. Outliers dapat dihilangkan dengan cara menormalisasikan data agar data memiliki range tertentu. Normalisasi dilakukan dengan menggunakan metode standar scaler.

```
 #NORMALISASI DATA  
data_train = StandardScaler().fit_transform(data_train)  
data_train
```

Output dari kode di atas yaitu sebagai berikut.

```
array([[ 1.73443926, -1.1075202 ,  0.64132669, ...,  0.11380926,  
        0.29460309,  1.07867028],  
       [ 0.57931213,  0.90291807, -0.03199781, ..., -0.58856355,  
        0.0796127 ,  0.50487408],  
       [ 0.4577198 , -1.1075202 ,  0.15653305, ...,  0.4228533 ,  
       -0.52087767,  1.0570176 ],  
       ...,  
       [-0.63661117, -1.1075202 , -0.00506483, ...,  0.67570751,  
       -4.7094834 , -3.3926096 ],  
       [-0.87979583,  0.90291807, -0.30132761, ...,  1.34998541,  
        0.42063193,  0.92710147],  
       [ 1.91682776,  0.90291807, -0.16666271, ..., -0.89760759,  
        0.14633386,  0.18008377]])
```

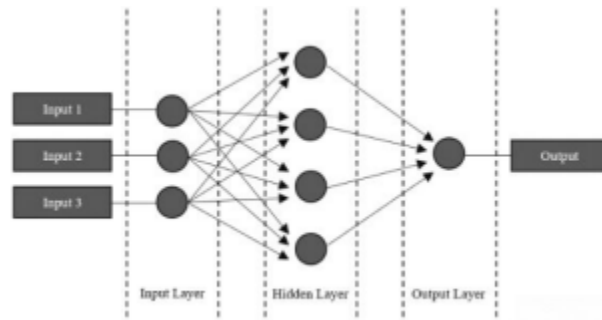
Standar scaler dilakukan untuk menormalisasi data agar data yang digunakan tidak memiliki penyimpangan yang besar sehingga outliers pada beberapa atribut akan hilang.

BAB 3

ALGORITMA ANN

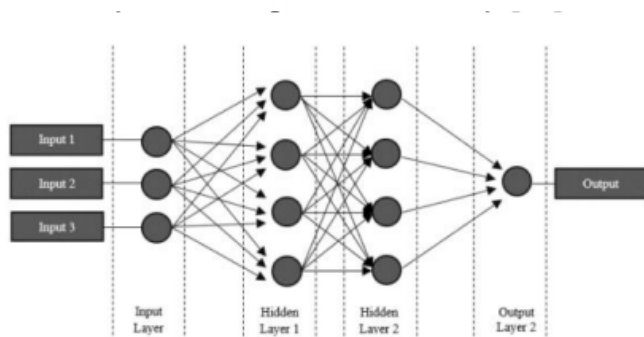
3.1 Algoritma ANN MLP

Algoritma ANN (Artificial Neural Network) merupakan metode deep learning yang bekerja dengan sistem kerja otak manusia dalam menyelesaikan suatu masalah. ANN mempelajari data pada masa lalu untuk memberikan keputusan terhadap data yang akan datang. ANN bekerja dengan algoritma untuk menginterpretasikan data non linear dari pola sekuensial. ANN memiliki arsitektur yang nantinya akan didistribusikan secara paralel dengan jumlah node yang banyak. Hubungan antar node disebut dengan *neutron*. Hubungan antar *neutron* yang memiliki nilai disebut dengan bobot sehingga setiap *neutron* memiliki nilai yang terkait sebagai nilai aktivasi neuron.



Pada masukan *artificial neuron*, input memiliki bobot setiap nilai input dikalikan dengan bobot individual. Setelah itu terdapat penjumlahan antara input dan bias. Kemudian penjumlahan bobot dan bias melewati fungsi aktivasi kemudian diteruskan ke output *neural network*.

MLP (Multi-layer perceptron) merupakan salah satu metode yang sederhana guna mengenalkan pola, terdiri dari *neutron* yang diurutkan menjadi lapisan. Lapisan pertama disebut lapisan input, lapisan terakhir disebut lapisan output, dan lapisan antara input dan output disebut hidden layer.



3.2 Proses Pembangunan Model

3.2.1 Proses Split Data

3.2.1.1 Proses Memisahkan Data X dengan Data Target

```
[143] data_train = data.copy()

#memisahkan data x dengan target (data y)
data_train.drop(data_train.columns[-1], axis = 1, inplace = True)

#menentukan target
y_target = data['class']
```

Dataset bernama data merupakan dataset sekumpulan data x dengan target. Pada proses ini kita akan memisahkan data x dengan data target. Data target adalah data dengan atribut class. Dengan dilakukannya proses ini, mempermudah kita dalam proses split data train dengan data test.

3.2.1.2 Proses Split Data Train dengan Data Test

```
▶ x = data_train
  y = y_target

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=42)
```

Proses split menggunakan library sklearn. Proses ini memiliki tujuan untuk membagi data menjadi dua bagian yaitu data train dan data testing. Data train digunakan untuk melatih model sedangkan data testing digunakan untuk mengevaluasi model yang telah dibangun.

3.2.2 Proses Membangun Model ANN

```
[146] ann = keras.Sequential([
    keras.layers.Dense(32, activation='sigmoid'),
    keras.layers.Dense(32, activation = "sigmoid"),
    keras.layers.Dense(64, activation='sigmoid'),
    keras.layers.Dense(64, activation = "sigmoid")
])
```

Proses membangun model dibantu dengan library keras dengan menggunakan activation sigmoid pada layer. Setelah kita membangun model, model tersebut kita compile seperti gambar di bawah ini.

```
▶ ann.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.05), loss = "sparse_categorical_crossentropy", metrics=['accuracy'])
```

3.2.3 Proses Training

Setelah model berhasil dibangun, tahap selanjutnya adalah fitting model pada data train. Hal ini dilakukan dengan tujuan untuk melatih model agar dapat melihat pola dari data yang kita miliki.

```
#fitting model  
train_data = ann.fit(x_train, y_train, batch_size= 32, epochs = 60)
```

BAB 4

EVALUASI MODEL

Dari proses training yang telah dilakukan, dengan memberikan nilai learning rate sebesar 0,05 disertai epochs sebanyak 60. Maka diperoleh hasil evaluasi pada data train sebagai berikut.

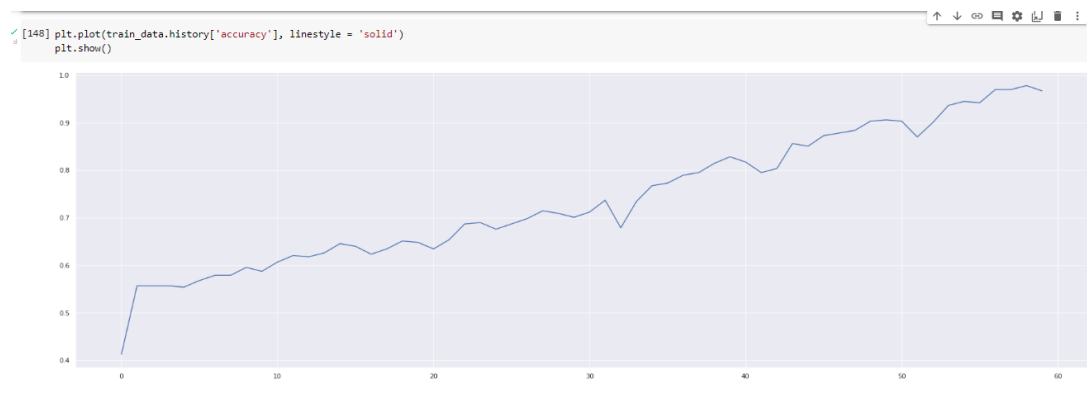
```
Epoch 47/60
12/12 [=====] - 0s 3ms/step - loss: 0.2866 - accuracy: 0.8781
Epoch 48/60
12/12 [=====] - 0s 3ms/step - loss: 0.2848 - accuracy: 0.8837
Epoch 49/60
12/12 [=====] - 0s 4ms/step - loss: 0.2395 - accuracy: 0.9030
Epoch 50/60
12/12 [=====] - 0s 3ms/step - loss: 0.2440 - accuracy: 0.9058
Epoch 51/60
12/12 [=====] - 0s 3ms/step - loss: 0.2582 - accuracy: 0.9030
Epoch 52/60
12/12 [=====] - 0s 3ms/step - loss: 0.3231 - accuracy: 0.8698
Epoch 53/60
12/12 [=====] - 0s 3ms/step - loss: 0.2747 - accuracy: 0.9003
Epoch 54/60
12/12 [=====] - 0s 3ms/step - loss: 0.1865 - accuracy: 0.9363
Epoch 55/60
12/12 [=====] - 0s 3ms/step - loss: 0.1707 - accuracy: 0.9446
Epoch 56/60
12/12 [=====] - 0s 3ms/step - loss: 0.1666 - accuracy: 0.9418
Epoch 57/60
12/12 [=====] - 0s 4ms/step - loss: 0.1032 - accuracy: 0.9695
Epoch 58/60
12/12 [=====] - 0s 3ms/step - loss: 0.1033 - accuracy: 0.9695
Epoch 59/60
12/12 [=====] - 0s 3ms/step - loss: 0.0979 - accuracy: 0.9778
Epoch 60/60
12/12 [=====] - 0s 3ms/step - loss: 0.0974 - accuracy: 0.9668
```

Akurasi pada data train yaitu sebesar 0.9668 dan juga error yang kecil yaitu sebesar 0.0974 . Sedangkan akurasi pada data testing yaitu sebesar 0,4725.

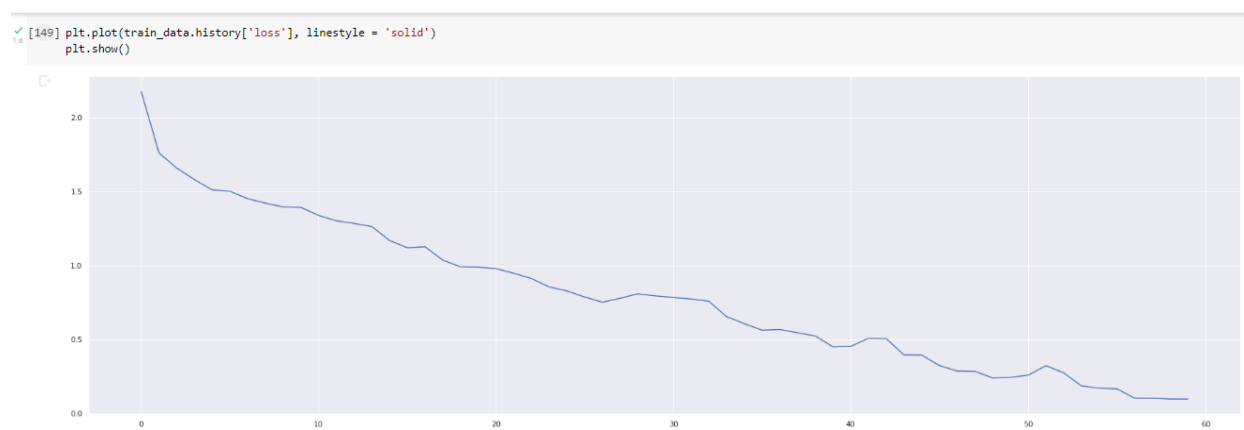
```
[150] print("Melakukan test pada model")
      results = ann.evaluate(x_test, y_test, batch_size=256)
      print("test loss, test acc:", results)
```

```
Melakukan test pada model
1/1 [=====] - 0s 209ms/step - loss: 3.7267 - accuracy: 0.4725
test loss, test acc: [3.7267351150512695, 0.47252747416496277]
```

Berikut merupakan grafik akurasi pada setiap iterasi.



Grafik di atas menandakan bahwa akurasi data train pada setiap iterasi akan mengalami peningkatan. Sedangkan pada grafik error akan mengalami penurunan pada setiap iterasi. Hal tersebut dapat dilihat pada grafik di bawah ini.



LAMPIRAN

LINK GOOGLE COLAB

https://colab.research.google.com/drive/1rCypL_sgUQHMneGeP_oQ-4QFLXI1CzBb?usp=sharing

LINK VIDEO PENJELASAN

https://drive.google.com/file/d/1zjpzP_fpmtcSCTIiewNfFazi6fD40-Pe/view?usp=sharing