Laporan Case Based-1

(CII3C3) Pembelajaran Mesin

oleh

Herman Gemilang – 1301204014

Kode dosen:

BDP

INFORMATIKA FAKULTAS

INFORMATIKA

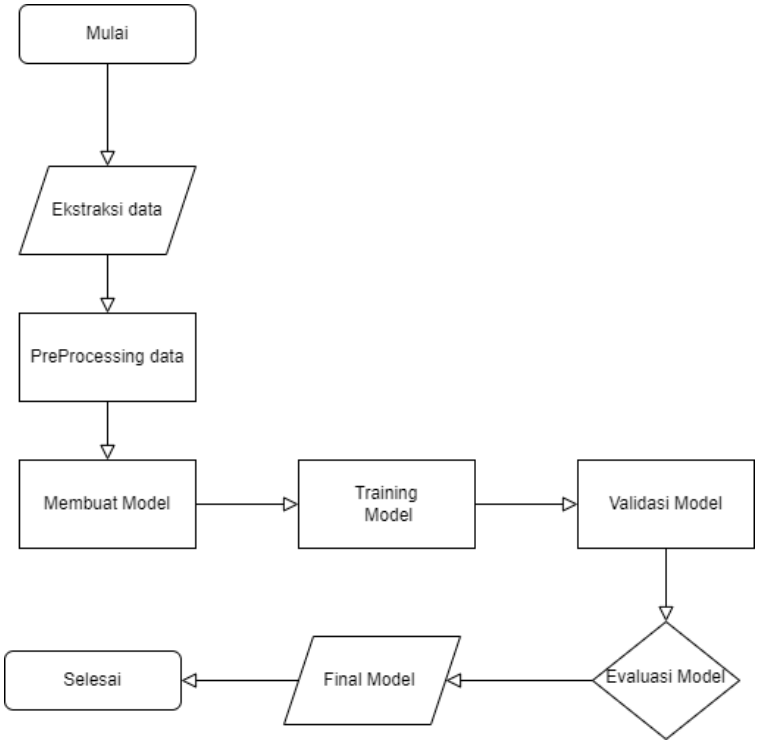
UNIVERSITAS TELKOM

2022

Ikhtisar

1.1 Pendahuluan

Dalam proses pembuatan model kali ini , penulis menggunakan metode ANN(Artificial Neural Network) untuk proses pengklasifikasian penyakit arrhythmia pada *dataset* arrhythmia Tahap pertama yang dilakukan adalah proses ekstraksi data mentah dari dataset, tahap kedua dilakukan preprocessing data yang meliputi handling missing value, filtering, dll, kemudian proses implementasi model , melakukan evaluasi model untuk melakukan perbaikan pada model. Diagram alur program yang akan dirancang adalah sebagai berikut.



1.2 Datasets

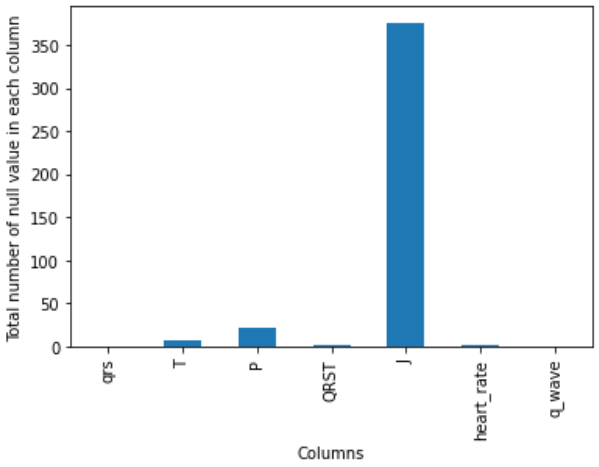
Pada tugas ini, *dataset* yang digunakan adalah data_arrhythmia.csv, berjumlah 452 records dan 280 atribut. Berikut adalah example data dari dataset tersebut.

1.3 Overview Datasets

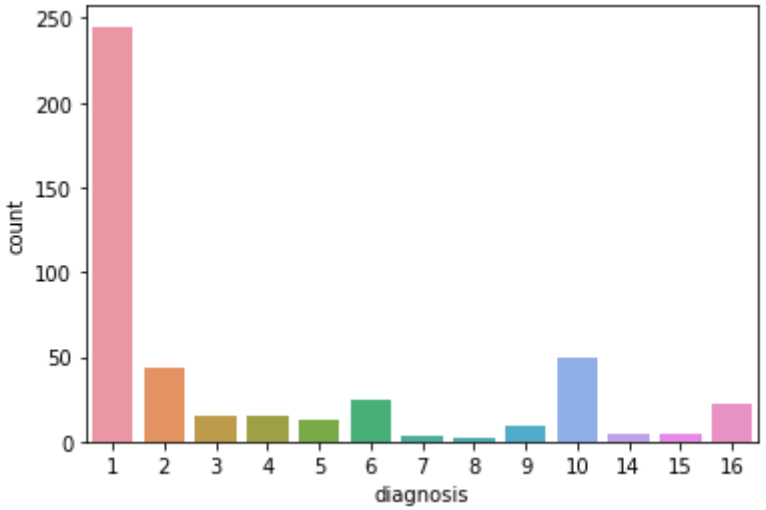
Pada datasets yang diberikan penulis melakukan proses identifikasi atribut. Dapat dilihat bahwa semua atribut yang ada di dalam dataset merupakan hal penting untuk melakukan klasifikasi.

Preprocessing data

Preprocessing data merupakan suatu proses menganalisa dan mengidentifikasi serta filtering suatu data mentah menjadi data siap pakai. Preprocessing data dilakukan dengan membersihkan records data null/kosong, melakukan proses identifikasi statistika. Hal yang pertama dilakukan adalah mencari dan menganalisis terlebih dahulu data mana saja yang banyak terdapat null data.



Selanjutnya saya Melihat besaran persentase dari atribut diagnosis yang akan menjadi parameter untuk output.



Class code : Class :

01	Normal
02	Ischemic changes (Coronary Artery Disease)
03	Old Anterior Myocardial Infarction
04	Old Inferior Myocardial Infarction
05	Sinus tachycardy
06	Sinus bradycardy
07	Ventricular Premature Contraction (PVC)
08	Supraventricular Premature Contraction

09	Left bundle branch block
10	Right bundle branch block
11	1. degree AtrioVentricular block
12	2. degree AV block
13	3. degree AV block
14	Left ventricule hypertrophy
15	Atrial Fibrillation or Flutter
16	Others

Langkah selanjutnya yang penulis gunakan dalam proses preprocessing data adalah melakukan proses penggantian nilai null menjadi nilai 0.

```
#mengganti nilai data NaN menjadi 0
new_data =new_data.replace(to_replace = np.nan, value = 0)
```

Dalam prosesnya penulis mengganti output yang semula 16 inputan yang berasal dari jumlah class “Diagnosis” menjadi 1 inputan yang terbagi menjadi binary classification yang terdiri dari:

1. 0 => [punya masalah jantung] = class “Diagnosis” 2-16
2. 1 => [Sehat] = class “Diagnosis” 1

```
4] target.replace(to_replace = 2, value = 0,inplace = True)
   target.replace(to_replace = 3, value = 0,inplace = True)
   target.replace(to_replace = 4, value = 0,inplace = True)
   target.replace(to_replace = 5, value = 0,inplace = True)
   target.replace(to_replace = 6, value = 0,inplace = True)
   target.replace(to_replace = 7, value = 0,inplace = True)
   target.replace(to_replace = 8, value = 0,inplace = True)
   target.replace(to_replace = 9, value = 0,inplace = True)
   target.replace(to_replace = 10, value = 0,inplace = True)
   target.replace(to_replace = 11, value = 0,inplace = True)
   target.replace(to_replace = 12, value = 0,inplace = True)
   target.replace(to_replace = 13, value = 0,inplace = True)
   target.replace(to_replace = 14, value = 0,inplace = True)
   target.replace(to_replace = 15, value = 0,inplace = True)
   target.replace(to_replace = 16, value = 0,inplace = True)
   target.replace(to_replace = 17, value = 0,inplace = True)
```

Terakhir adalah melakukan split data yang terdiri dari X_train,X_test,Y_train,Y_test yang akan digunakan untuk proses training dan testing.

```
#proses split data train dan testing
X_train,X_test,Y_train,Y_test = train_test_split( final_data, target, test_size=0.2,  random_state = 1)
Y_train
```

```
272    0
101    1
379    0
448    0
230    1
      ..
255    1
72     1
396    1
235    0
37     0
Name: diagnosis, Length: 361, dtype: int64
```

Penerapan Algoritma

3.1 Rancangan Arsitektur Model

Arsitektur model Artificial Neural Networks (ANN) yang diusulkan pada tugas ini terdiri dari input layer, Flatten, Dense, Dropout, dan sigmoid.

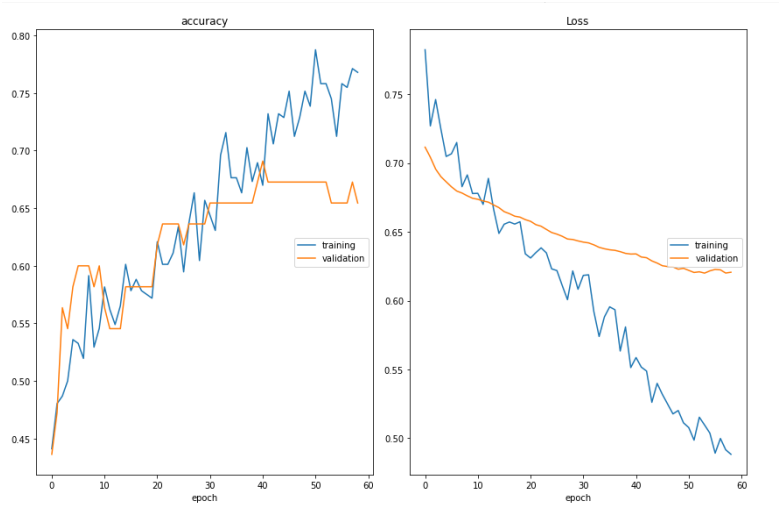
```
#proses intialisasi data

model = tf.keras.models.Sequential()

#add a input layer
model.add(keras.layers.Dense(units=15, activation='relu', input_shape=X_train[0].shape))

#Add a hidden layer
model.add(keras.layers.Dense(units = 12,kernel_initializer ='glorot_uniform',name='Hidden-Layer-1',activation='relu'))
model.add(tf.keras.layers.Dropout(0.25))
model.add(keras.layers.Dense(units = 9,kernel_initializer ='glorot_uniform',name='Hidden-Layer-2',activation='relu'))
model.add(tf.keras.layers.Dropout(0.25))
#Add a output layer
model.add(keras.layers.Dense(units = 1,kernel_initializer ='glorot_uniform',activation='sigmoid'))

# compile the keras model/hyperparameter
model.compile(optimizer=tf.keras.optimizers.SGD(0.01), loss='binary_crossentropy',metrics=['accuracy'])
```

Gambar 3.2 Kodingan model ANN

3.2 Hyperparameter Tuning

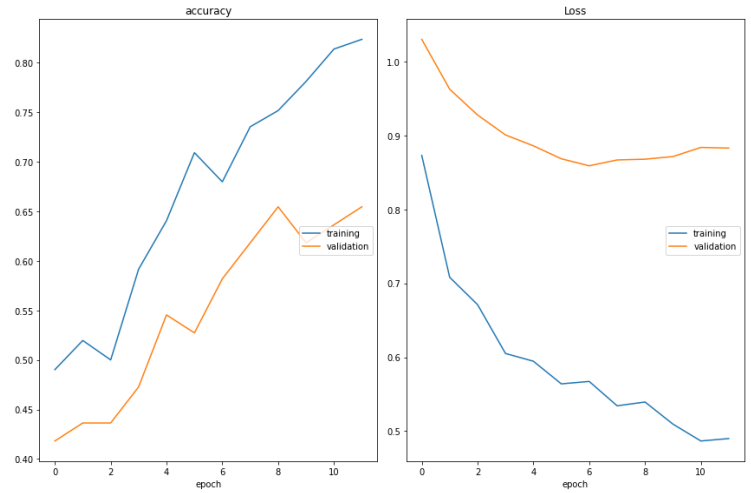
Parameter yang tepat untuk optimizer merupakan hal yang sangat penting bagi proses dan hasil klasifikasi. Hyperparameter Tuning memiliki peran penting dalam pembelajaran mesin karena parameter yang dihasilkan mempengaruhi kinerja model ANN secara signifikan. Hyperparameter Tuning disarankan penggunaannya dalam proses Model ANN untuk mengoptimalkan pemrosesan dataset pada Model ANN yang dibentuk. Pada tugas ini, dilakukan metode hyperparameter tuning untuk menentukan parameter optimizer dan dropout. Pelatihan model pada saat hyperparameter dilakukan menggunakan epoch berjumlah 200. Adapun parameter optimizer yang digunakan adalah

Pembanding	Parameter
Optimizer	Adamax
Optimizer	SGD
Optimizer	RMSprop
Optimizer	Nadam



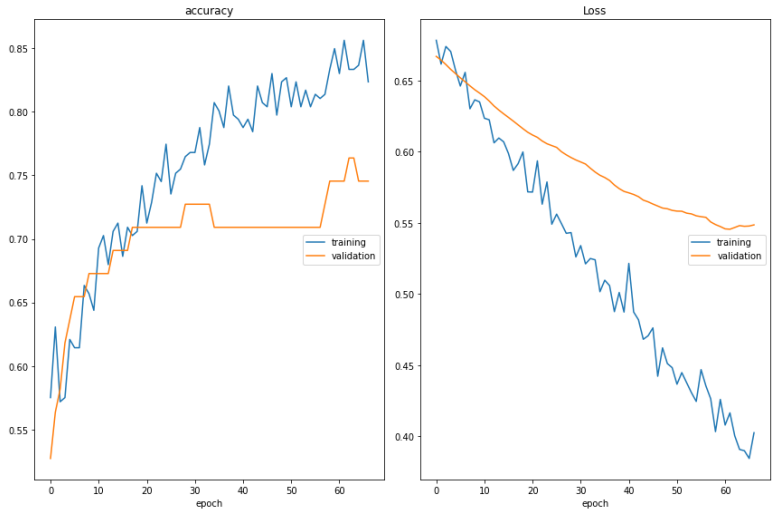

```
accuracy
        training                (min:    0.441, max:    0.788, cur:    0.768)
        validation              (min:    0.436, max:    0.691, cur:    0.655)
Loss
        training                (min:    0.488, max:    0.782, cur:    0.488)
        validation              (min:    0.620, max:    0.712, cur:    0.621)
```

Training & Val loss/acc) with SGD



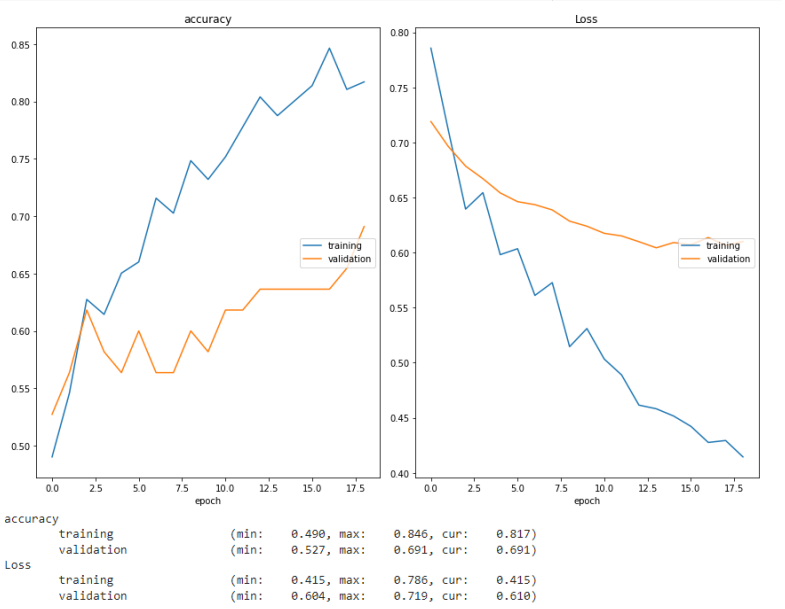

```
accuracy
        training                (min:    0.490, max:    0.824, cur:    0.824)
        validation              (min:    0.418, max:    0.655, cur:    0.655)
Loss
        training                (min:    0.487, max:    0.873, cur:    0.490)
        validation              (min:    0.859, max:    1.030, cur:    0.883)
```

(Training & Val loss/acc) with RMSprop




```
accuracy
        training                (min:    0.572, max:    0.856, cur:    0.824)
        validation              (min:    0.527, max:    0.764, cur:    0.745)
Loss
        training                (min:    0.384, max:    0.678, cur:    0.402)
        validation              (min:    0.546, max:    0.667, cur:    0.549)
```

(Training & Val loss/acc) with Adamax



(Training & Val loss/acc) with Adamax

Code untuk proses Hyperparameter Tuning

```
# compile the keras model/hyperparameter
model.compile(optimizer=tf.keras.optimizers.Nadam(0.001), loss='binary_crossentropy',metrics=['accuracy'])
```

Skenario	Optimize	Akurasi
1	**Nadam**	**80%**
2	RMSprop	71%
3	Adamax	70%
4	SGD	67%

3.3 Hasil Analisis

Berdasarkan hasil dari setiap skenario, Bisa kita lihat bahwa penggunaan hyperparameter dapat mengubah nilai akurasi. Namun, dari ke-4 skenario tersebut, skenario keempat lebih baik dalam akurasi, loss, precision, recall, f1-score. Dapat kita simpulkan bahwa untuk penggunaan optimizer dalam kasus ini yang terbaik adalah menggunakan Nadam, karena menghasilkan grafik yang selaras antara training dan validation (tidak overfitting / underfitting).

Evaluasi hasil

Proses selanjutnya adalah proses test/predict dari model tersebut. Model akan memprediksi sebanyak 91 data.

```
] #melihat jumlah kolom dan baris setelah proses split
  print(X_train.shape, Y_train.shape, X_test.shape, Y_test.shape)

  (361, 277) (361,) (91, 277) (91,)
```

```
✓ 0d  #menampilkan matriks akurasi data testing
      print(confusion_matrix(Y_test, y_pred))
      print("akurasi : ",accuracy_score(Y_test, y_pred)*100,'%')

      [[28 11]
       [ 5 47]]
      akurasi :  82.41758241758241 %
```

Proses Testing 91 data tersebut menghasilkan akurasi prediksi sebesar 82%. Setiap bentuk model dan bentuk hyperparameter sangat berpengaruh terhadap output yang diberikan machine learning. Metode Nadam merupakan metode terbaik yang saya temukan selama proses perancangan model ini. Penulis merasa model yang dibangun tidak optimal, hal ini dikarenakan beberapa faktor, diantaranya kurangnya pemahaman penulis mengenai karakteristik data, kesulitan dalam mengelola dataset, dll.