BAB 5

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

5.1.1 Batasan Implementasi

Batasan implementasi dari tugas akhir ini adalah menggunakan metode *Backpropagation Neural Network* (BPNN) untuk mengklasifikasi persalinan dengan variabel yaitu usia kehamilan, pelebaran serviks, kontraksi, pecah ketuban, usia ibu, jumlah kehamilan, dan komplikasi, tunggal/kembar.

5.1.2 Lingkungan Implementasi

Lingkungan implementasi sistem pada tugas akhir ini terdiri dari beberapa komponen pendukung yang berupa perangkat keras dan perangkat lunak.

1. Perangkat Keras:

   Spesifikasi yang digunakan pada perangkat keras yaitu sebagai berikut:

   a. *Processor* : Intel Dual-Core N3050 CPU @ 2.16GHz
   b. *Memory* : 4 GB
   c. *Hard disk* : 500 GB

2. Perangkat Lunak:

   Spesifikasi yang digunakan pada perangkat keras yaitu sebagai berikut:

   a. *Operating System* : *Windows* 10
   b. Bahasa Pemograman : *Python*
   c. *Browser* : *Google Chrome*
   d. *Tools* : *Google Colab*

5.1.3 Implementasi Sistem

Implementasi sistem adalah tahapan-tahapan *coding* dan hasil pada *Google Colab*.

1) Proses import dataset (disimpan dalam variabel *dataset* disimpan dalam *dataset),*

dimana variable target 1 diubah menjadi 0, karena

nantinya akan diubah menjadi vector supaya menghasilkan nilai untuk *precission* dan *recall*

```
[ ] # import dataset
    df = pd.read_excel('inibener.xlsx')
    df.head()
```

	usia kehamilan	pelebaran serviks	kontraksi	pecah ketuban	usia ibu	jumlah kehamilan	kompikasi	tunggal/kembar	target
0	31	3	1	2	26	1	2	1	1
1	28	8	1	2	25	1	2	2	1
2	31	3	2	2	28	2	2	1	1
3	27	2	2	2	27	2	0	2	1
4	28	6	2	2	17	1	2	1	2

```
[ ] df['target'] = df['target'].replace([1], 0)
    df.target.value_counts()
```

```
0    374
2     16
Name: target, dtype: int64
```

```
df['target'] = df['target'].replace([2], 1)
df.target.value_counts()
```

```
0    374
1     16
Name: target, dtype: int64
```

```
[ ] # dataset composition
    print('='*50)
    print("Initial shape of the dataset : ", df.shape)
    print('='*50)
    # check null and dataset type
    print(df.info())
    print('=' * 50)
```

```
[ ] # change to vector
    data_new = df.copy()
    X = np.array(data_new.drop(columns = ['target'], axis=1))
    y = np.array(data_new['target'])
```

2) Nilai target (*target values*) yang didapat adalah untuk target 0 yaitu 374 sedangkan untuk target 1 yaitu 16

```
df['target'] = df['target'].replace([2], 1)
df.target.value_counts()
```

```
0    374
1     16
Name: target, dtype: int64
```

3) Melakukan pembagian data latih dan data uji, dimana untuk pembagian data latih

dilakukan dengan cara membagi data keseluruhan yang berjumlah 390 data menjadi dua bagian yaitu 80% berjumlah 312 data latih dan 60% berjumlah 234 data latih. Sedangkan Pembagian data uji dilakukan dengan cara membagi data keseluruhan yang berjumlah 390 data menjadi dua bagian yaitu 20% berjumlah 78 data uji dan 40% berjumlah 156 data uji.

```
# splitting dataset
#ini untuk data uji
X_train, X_test, y_train, y_test = train_test_split(X,y , test_size = .8, random_state = 42)

# check split dataset
print('Train feature instances=', X_train.shape)
print('Test feature instances=', X_test.shape)
print('Train label instances=', y_train.shape)
print('Test label instances=', y_test.shape)
```

```
Train feature instances= (78, 8)
Test feature instances= (312, 8)
Train label instances= (78,)
Test label instances= (312,)
```

```
# splitting dataset
#ini untuk data uji
X_train, X_test, y_train, y_test = train_test_split(X,y , test_size = .6, random_state = 42)

# check split dataset
print('Train feature instances=', X_train.shape)
print('Test feature instances=', X_test.shape)
print('Train label instances=', y_train.shape)
print('Test label instances=', y_test.shape)
```

```
Train feature instances= (156, 8)
Test feature instances= (234, 8)
Train label instances= (156,)
Test label instances= (234,)
```

4) Menampilkan hasil dari data latih dan data uji kedalam tabel

```
[16] # display data training
     X_train_df.head()
```

	usia kehamilan	pelebaran serviks	kontraksi	pecah ketuban	usia ibu	jumlah kehamilan	kompikasi	tunggal/kembar
0	28	0	1	1	27	3	2	1
1	28	0	1	2	31	1	2	1
2	30	0	1	2	25	2	2	1
3	33	0	1	2	25	4	2	1
4	34	0	1	2	32	1	2	1

```
# display data testing
X_test_df.head()
```

	usia kehamilan	pelebaran serviks	kontraksi	pecah ketuban	usia ibu	jumlah kehamilan	kompikasi	tunggal/kembar
0	28	0	1	2	25	3	2	1
1	32	0	1	2	23	2	2	1
2	27	1	1	1	25	2	2	1
3	34	4	1	2	36	2	2	1
4	22	2	1	2	32	1	2	1

5) Melakukan pengulangan (*epoch*) sebanyak 100 kali.

```
[23] # train model sequentials with training dataset
     history = model.fit(X_train, y_train, batch_size=10, epochs=100, verbose = 'auto')

Epoch 72/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1372 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 73/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1385 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 74/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1392 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 75/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1354 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 76/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1359 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 77/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1357 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 78/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1382 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 79/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1388 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 80/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1361 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 81/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1364 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 82/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1349 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 83/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1352 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 84/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1342 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 85/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1335 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 86/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1355 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
```


[23] Epoch 84/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1342 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 85/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1335 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 86/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1355 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
```
Epoch 87/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1341 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 88/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1346 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 89/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1322 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 90/100
8/8 [==============================] - 0s 6ms/step - loss: 0.1356 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 91/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1338 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 92/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1317 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 93/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1317 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 94/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1327 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 95/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1329 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 96/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1318 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 97/100
8/8 [==============================] - 0s 5ms/step - loss: 0.1308 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 98/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1330 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 99/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1319 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
Epoch 100/100
8/8 [==============================] - 0s 4ms/step - loss: 0.1308 - accuracy: 0.9615 - precision: 0.0000e+00 - recall: 0.0000e+00
```


6) Melakukan pengujian menggunakan *confusion matrix*

```
[25] y_pred_awal = model.predict(X_test)

     print('='*40)
     print('Confusion Matrix')
     print('='*40)

     res = tf.math.confusion_matrix(
         y_test,
         y_pred_awal,
         num_classes=None,
         weights=None,
         dtype=tf.dtypes.int32,
         name=None
     )
     print(res)
```

```
10/10 [==============================] - 0s 2ms/step
========================================
Confusion Matrix
========================================
tf.Tensor(
[[299   0]
 [ 13   0]], shape=(2, 2), dtype=int32)
```

7) Didapatkan hasil akurasi sebesar 97 %

```
3/3 [==============================] - 0s 3ms/step
Results for Multilayer Perceptron Model
Akurasi =  0.9743589743589743
                    precision    recall  f1-score   support

                 0       0.97      1.00      0.99        76
                 1       0.00      0.00      0.00         2

          accuracy                           0.97        78
         macro avg       0.49      0.50      0.49        78
      weighted avg       0.95      0.97      0.96        78
```

5.2 Data Balance dan Imbalance

Pada penelitian ini komposisi dataset tidak seimbang dimana kelas kelas premature sebanyak 16 dan kelas normal sebanyak 374. Pada model *Backpropagation* menggunakan dataset tidak seimbang akan menyebabkan bias terhadap kelas yaitu model akan mendeteksi kelas yang lebih besar. Pada data imbalance menggunakan arsitektur jaringan 8-12-1, learning rate 0,01 dengan pembagian dataset 80;20 dan maksimal epoch 100. Adapun metric yang digunakan adalah f1-score. Berikut merupakan hasil testing f1-score pada data imbalance

Tabel 5.4 Hasil testing f1-score pada data imbalance

Label	Precision	Recall	F1-score
0	0,97	1,00	0,98
1	0,00	0,00	0,00

Tabel 5.1 memperlihatkan bahwa :

1. Nilai presisi pada kelas 0 menjelaskan bahwa berapa rasio persalinan normal dari keseluruhan persalinan yang diprediksi normal adalah 0,97, sedangkan nilai presisi

pada kelas 1 menjelaskan bahwa berapa rasio persalinan prematur dari keseluruhan putusan yang diprediksi normal adalah 0,00.

2. Nilai recall pada kelas 0 menjelaskan bahwa berapa rasio persalinan normal dari keseluruhan persalinan yang diprediksi normal adalah 1,00, sedangkan nilai recal pada kelas 1 menjelaskan bahwa berapa rasio persalinan prematur dari keseluruhan persalinan yang diprediksi normal adalah 0,00
3. Nilai F1-score pada kelas 0 menjelaskan bahwa berapa rasio persalinan normal dari keseluruhan persalinan yang diprediksi normal adalah 0,98, sedangkan nilai recal pada kelas 1 menjelaskan bahwa berapa rasio persalinan prematur dari keseluruhan persalinan yang diprediksi normal adalah 0,00

Dapat disimpulkan bahwa nilai untuk setiap kelas sangat jauh nilainya dimana untuk nilai kelas normal 0,98 dan kelas preamaturnya 0,00 adapun value loss nya 0,1954 dan akurasinya 0,9530 dengan f1-score yang compang maka diperlukan penyeimbangan dataset. Penyeimbangan dataset dilakukan dengan cara membagi komposisi dataset dengan 1 : 2 bobot komposisi dataset 100 untuk persalinan normal dan 55 untuk persalinan premature, berikut merupakan hasil testing f1-score pada data Balance

Tabel 5.5 Hasil testing f1-score pada data Balance

Label	Precision	Recall	F1-score
0	0,48	0,50	0,49
1	0,94	0,97	0,95

Tabel 5.2 memperlihatkan hasil data testing dengan f1-score untuk data yang balance cukup meningkat dengan nilai untuk kelas normal 0,49 dan kelas premature 0,95 dan value losnya 0,1954 dan hasil akurasinya 0,97. Terlihat bahwa model klasifikasi sudah dapat mengklasifikasikan kelas premature maka dataset balance akan dilakukan untuk pengujian berikutnya

5.3 Sensitivitas parameter uji

Setelah dilakukan penyeimbangan dataset maka dilakukan beberapa parameter untuk melihat performa akurasi yang lebih baik, adapun beberapa parameter yang diujikan dengan arsitektur jaringan dengan layer 8-12-1 dan 8-16-1, learning rate 0,01,dan 0,03 dan 0,9 maksimal epoch 100, dan pengujian 80:20 dan 60:40

5.3.3 Pengujian Akurasi 80%:20%

Pengujian dengan menggunakan 80% data latih dan 20% data uji dengan menggunakan , arsitektur jaringan dengan layer 8-12-1 dan 8-16-1, learning rate 0,01,0,03, dan 0,9 dan *epoch* 50, 100, dan 200. Dapat dilihat pada tabel 5.6

Tabel 5.6 pengujian akurasi *Epoch 50*

No	Arsitektur jaringan	Learning rate	*Epoch*	Akurasi
1	8-12-1	0,1	50	95,51 %
2		0,3	50	95,51 %
3		0,9	50	95,51 %
4	8-16-1	0,1	50	95,51 %
5		0,3	50	95,51 %
6		0,9	50	95,51 %

Tabel 5.6 menjelaskan hasil testing untuk pembagian data 80:20 yang mana pada arsitektur 8-12-1 dan 8-12-1 dengan learning rate 0,01 dengan pengujian *Epoch* 50 menghasilkan akurasi yang sama yaitu 95,51 %

Tabel 5.7 pengujian akurasi *Epoch 100*

No	Arsitektur jaringan	Learning rate	*Epoch*	Akurasi
1	8-12-1	0,1	100	95,51 %
2		0,3	100	95,51 %
3		0,9	100	95,51 %
4	8-16-1	0,1	100	95,51 %
5		0,3	100	95,51 %
6		0,9	100	95,51 %

Tabel 5.7 menjelaskan hasil testing untuk pembagian data 80:20 yang mana pada arsitektur 8-12-1 dengan learning rate 0,1 dengan pengujian *Epoch* 100 menghasilkan akurasi yang sama yaitu 95,51 %

Tabel 5.8 pengujian akurasi *Epoch 200*

No	Arsitektur jaringan	Learning rate	*Epoch*	Akurasi

1	8-12-1	0,1	200	95,51 %
2		0,3	200	95,51 %
3		0,9	200	95,51 %
4	8-16-1	0,1	200	95,51 %
5		0,3	200	95,51 %
6		0,9	200	95,51 %

Tabel 5.8 menjelaskan hasil testing untuk pembagian data 80:20 yang mana pada arsitektur 8-16-1 dengan learning rate 0,03 dengan pengujian *Epoch* 200 menghasilkan akurasi yang sama yaitu 95,51 %.

Dari hasil pengujian akurasi 80% : 20 %, arsitektur jaringan [8-12-1] dan [8-16-1], *learning rate* (0,1, 0,3, 0,9), pengujian *Epoch* 50,100,200 menghasilkan nilai akurasi yang sama yaitu 95,51 %

5.3.4 Pengujian Akurasi 60%:40%

Pengujian dengan menggunakan 80% data latih dan 20% data uji dengan menggunakan, arsitektur jaringan dengan layer 8-12-1 dan 8-16-1, learning rate 0,01,0,03, dan 0,9 dan *epoch* 50, 100, dan 200. Dapat dilihat pada tabel 5.3

Tabel 5.9 pengujian akurasi *Epoch 50*

No	Arsitektur jaringan	Learning rate	*Epoch*	Akurasi
1	8-12-1	0,1	50	96,15 %
2		0,3	50	95,30 %
3		0,9	50	95,30 %
4	8-16-1	0,1	50	96,15 %
5		0,3	50	94,44 %
6		0,9	50	95,30 %

Tabel 5.9 menjelaskan hasil testing untuk pembagian data 60:40 yang mana pada arsitektur [8-16-1] dan [8-12-1] dengan learning rate 0,3, pengujian *Epoch* 50 menghasilkan akurasi terkecil yaitu 94,44 %

Tabel 5.60 pengujian akurasi *Epoch 100*

No	Arsitektur jaringan	Learning rate	*Epoch*	Akurasi
1	8-12-1	0,1	100	94,87 %
2		0,3	100	95,73 %
3		0,9	100	95,30 %
4	8-16-1	**0,1**	**100**	**96,58 %**
5		0,3	100	95,73 %
6		0,9	100	95,30 %

Tabel 5.60 menjelaskan hasil testing untuk pembagian data 60:40 yang mana pada arsitektur 8-12-1 dan 8-16-1 dengan learning rate 0,1 dengan pengujian *Epoch* 100 menghasilkan akurasi terkecil yaitu 94,87 %

Tabel 5.61 pengujian akurasi *Epoch 200*

No	Arsitektur jaringan	Learning rate	*Epoch*	Akurasi
1	8-12-1	**0,1**	**200**	**96,58 %**
2		0,3	200	94,44 %
3		0,9	200	95,30 %
4	8-16-1	0,1	200	96,15 %
5		0,3	200	95,30 %
6		0,9	200	95,30 %

Tabel 5.61 menjelaskan hasil testing untuk pembagian data 60:40 yang mana pada arsitektur 8-12-1 dan 8-16-1 dengan learning rate 0,1 dengan pengujian *Epoch* 200 menghasilkan akurasi terkecil yaitu 96,15 %.

Dari hasil pengujian akurasi 60% : 40 %, arsitektur jaringan [8-12-1] dan [8-16-1], *learning rate* (0,1, 0,3, 0,9), pengujian *Epoch* 50,100,200 menghasilkan nilai akurasi yang tinggi yaitu pada arsitektur jaringan [8-12-1] dan [8-16-1], *learning rate* (0,01), pengujian *epoch* (100 dan 200) menghasilkan akurasi tertinggi yaitu 96,58 %