BAB IV
HASIL DAN PEMBAHASAN

4.1 Pemahaman Data

Pemahaman data adalah proses mendapatkan informasi dari kumpulan data yang ada. Data yang digunakan pada penelitian ini mengunakan data dari NASA POWER. Data yang diperoleh dari NASA POWER merupakan data cuaca tahun 2012 sampai dengan 2022. Data tersebut mencakup 8 paramater yang digunakan yaitu curah hujan,suhu maksimum,suhu minimum, kelembaban spesifik, kelembaban relatif, kecepatan angin maksimum, kecepatan angin minimum dan tekanan permukaan. Data ini memiliki format csv (*comma separated values*) terdiri 3743 sampel data.

Tabel 4.1 Dataset NASA POWER

Year	Mo	Dy	Suhu Max	Suhu Min	Kelembaban Spesifik	Kelembaban Relatif	Curah Hujan	Tekanan Permukaan	Kecepatan Angin Max	Kecepatan Angin Min
2012	1	1	30.88	22.43	18.19	85.00	19.37	100.90	2.37	1.10
2012	1	2	27.55	24.21	19.10	93.06	20.39	100.95	2.92	0.95
2012	1	3	28.36	23.41	19.17	92.81	10.95	100.92	2.91	1.38
2012	1	4	29.31	23.92	19.17	90.25	2.07	100.96	2.30	1.19
2012	1	5	28.80	24.06	19.17	91.19	3.66	100.97	2.95	1.39
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
2022	3	27	29.50	25.13	19.84	88.62	4.37	100.88	1.56	0.75
2022	3	28	29.45	24.57	19.71	89.81	0.60	100.81	2.69	1.12
2022	3	29	30.24	24.44	19.65	87.88	0.02	100.78	2.09	1.08
2022	3	30	29.94	24.74	19.59	86.44	0.49	100.76	1.69	0.17
2022	3	31	28.97	24.90	19.35	86.44	9.03	100.86	1.48	0.54

Tabel 4.2 Kategori Hujan BMKG

No	Kategori	Curah Hujan (mm/hari)
1	Hujan	>= 0.5
2	Tidak Hujan	< 0.5

Kategori curah hujan dikategorikan menjadi berawan/tidak hujan dan hujan BMKG. Pada data tersebut digunakan sebagai label dengan satuan milimiter (mm)

4.2 Data *Preprocessing*

Preprocessing data adalah bagian penting dari pemahaman data yang dapat memengaruhi keakuratan data yang akan digunakan. *Preprocessing* data ini dilakukan dalam beberapa tahap, pertama dimulai dari Eksplorasi Data Analisis (EDA), pembersihan data, untuk mentransformasikan data mentah sebelum melalui proses pemodelan atau analisis dan pengelompokkan data *training* dan data *testing.*

4.2.1. Eksplorasi Data Analisis (EDA)

Eksplorasi data merupakan proses penyelidikan awal terhadap data untuk memahami pola, karakterisitik dan informasi. Eksplorasi data ini tahapan penting dalam analisis data karena membantu dalam memberikan informasi. Pada eksplorasi data ini dilakukan *input* dataset terlebih dahulu dengan menggunakan format csv (*comma separated values*). Pada tabel 4.3 Deskripsi Statistik Data berikut ini maka eksplorasi data dapat dipahami.

Tabel 4.3 Deskripsi Statistik Data

	Suhu Max	Suhu Min	Kelembaban Spesifik	Kelembaban Relatif	Curah Hujan	Tekanan Permukaan	Kecepatan Angin Max	Kecepatan Angin Min
Count	3743	3743	3743	3743	3743	3743	3743	3743
Mean	29.89	23.51	18.77	88.20	7.01	100.86	2.45	0.97
Std	1.80	0.87	0.82	3.42	14.29	0.12	0.75	0.43

Min	25.57	19.80	14.34	69.88	0.00	100.41	0.91	0.02
25%	29.02	23.02	18.25	86.25	0.39	100.78	1.84	0.63
50%	29.94	23.60	18.80	88.56	2.52	100.86	2.38	1.09
75%	30.74	24.12	19.35	90.59	8.63	100.95	2.97	1.33
Max	35.73	25.86	21.55	96.06	508.85	101.31	5.92	1.95

4.2.2. Pembersihan Data

Pembersihan data adalah proses mengidentifikasi, memeriksa dan membersihkan data yang tidak relevan dari suatu dataset. Tujuan pembersihan data ini adalah memastikan keakuratan dan kualitas data supaya dapat digunakan dalam analisis data. Pada tabel 4.4 berikut hasil dari pemeriksaan data pada kolom tabel menunjukkan ada atau tidak adanya jumlah data yang hilang pada setiap variabelnya. Jika ada nilai yang hilang maka dapat dilakukan dengan nilai rata-rata dari variabel sebagai penggantian nilai. Pada pembersihan data ini terlihat juga ada beberapa parameter yang tidak perlukan yaitu tahun, bulan dan hari maka di lakukan penghapusan untuk variabel tersebut.

Tabel 4.4 Hasil Pemeriksaan Data

Parameter	Jumlah Data Hilang
Year	0
Mo	0
Dy	0
Suhu Max	0
Suhu Min	0
Kelembaban Spesifik	0
Kelembaban Relatif	0
Curah Hujan	0

Tekanan Permukaan	0
Kecepatan Angin Max	0
Kecepatan Angin Min	0

4.2.3. Data *Transormation*

Tujuan dari data *transformation* adalah untuk meningkatkan kualitas data yang dapat mengubah data mentah menjadi bentuk yang lebih terstruktur dan siap untuk pemodelan lebih lanjut. Tranformasi data ini dilakukan pada dataset di kolom curah hujan dalam bentuk angka pada tabel 4.5 berikut, kemudian data angka pada kolom curah hujan tersebut diubah menjadi label dengan kategori hujan dan tidak hujan. Probabilistik curah hujan oleh BMKG dengan intensitas hujan >=0,5 mm/hari dan tidak hujan dengan nilai <0,5 mm/hari.

Berdasarkan hasil dari transformasi data, ditemukan label hujan sebanyak 2722 sedangkan label tidak hujan sebanyak 1021. Setelah ditemukan hasil dari transformasi data tersebut, maka label diubah ke dalam bentuk angka agar proses prediksi mendapatkan hasil yang lebih akurat. Label “Hujan” diganti menjadi angka “1”, dan label “Tidak Hujan” diganti menjadi angka “0”.

Tabel 4.5 Transformasi Data

Curah Hujan		Label Hujan
16.37		Hujan
20.39		Hujan
20.39		Hujan
20.95		Hujan
2.07		Hujan
3.66		Hujan
.....	Transformasi data >>>	.....

4.37		Hujan
0.60		Hujan
0.02		Tidak Hujan
0.49		Tidak Hujan
9.03		Hujan

4.2.4. *Spliting* Data

Spliting data adalah proses membagi dataset menjadi subset yang berbeda untuk tujuan pengembangan model,evaluasi kinerja,dan pengujian.

Sebelum proses pembagian data, data x digunakan sebagai prediktor, yang terdiri dari delapan parameter yang digunakan dalam penelitian, dan data y digunakan sebagai target yang terdiri dari variabel label hujan. Selanjutnya, dilakukan proses pembagian data, yang terdiri dari x_train, x_test, y_train, dan y_test (Gambar 4.1). Proses pembagian data *train:test* 50%:50% dengan *shuffle=false*, yang menunjukkan bahwa data tidak diacak (data asli). Dimana pada proses ini ditemukan ada sebanyak 2994 data train dan 749 data test.

```
[ ]  from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
     print("Jumlah Data X_train = ",X_train.shape)
     print("Jumlah Data Y_train = ",y_train.shape)
     print("Jumlah Data X_test = ",X_test.shape)
     print("Jumlah Data Y_test = ",y_test.shape)

     Jumlah Data X_train =  (2994, 8)
     Jumlah Data Y_train =  (2994, 1)
     Jumlah Data X_test =  (749, 8)
     Jumlah Data Y_test =  (749, 1)
```

Gambar 4.1 *Spliting* Data

4.3. Model *Recurrent Neural Network* (RNN)

Model RNN (*Recurrent Neural Network*) adalah jenis model jaringan saraf tiruan yang dirancang khusus untuk mengolah data urutan atau data yang memiliki komponen waktu. RNN sangat cocok untuk tugas-tugas seperti pemprosesan bahasa alami, pemodelan bahasa, translasi mesin, dan prediksi urutan. Seperti halnya Model ANN (*Artificial Neural Network*), Model RNN juga dapat dibangun

menggunakan metode sequential yang terdapat di library keras *tensorflow*. Namun, ada perbedaan dalam arsitektur dan cara kerja antara Model ANN dan Model RNN.

Model RNN memiliki struktur rekursif yang memungkinkan informasi untuk dijalankan melalui lapisan yang berbeda dalam rangkaian waktu. Dalam Model RNN, setiap neuron di lapisan tersembunyi memiliki sambungan ke neuron itu sendiri pada waktu sebelumnya. Ini memungkinkan RNN untuk menyimpan dan mempertimbangkan informasi kontekstual dari waktu sebelumnya saat memproses input pada waktu sekarang. Model yang diberikan adalah model jaringan saraf rekurensi (RNN) yang menggunakan *layer* LSTM (*Long Short-Term Memory*) dan *layer Dropout*. Berikut adalah penjelasan singkat tentang model tersebut:

Sequential Model, model kerangka kerja yang digunakan untuk membangun model saraf jaringan secara sekuensial, artinya lapisan-lapisan ditambahkan satu per satu. LSTM *layer* 1 adalah lapisan LSTM pertama dalam model. Lapisan ini memiliki 5 unit (neuron) dan menggunakan *argument return_sequences=True*, yang berarti menghasilkan keluaran berurutan yang diteruskan ke lapisan berikutnya. *Input* yang diterima oleh lapisan ini adalah dengan bentuk (jumlah_timestep, 1). *Dropout* layer 1 adalah lapisan *Dropout* yang digunakan untuk menghindari *overfitting*. *Dropout* secara acak mengabaikan sebagian unit pada lapisan sebelumnya dengan level *dropout* sebesar 0.2. LSTM *layer* 2, lapisan LSTM kedua dalam model. Seperti lapisan sebelumnya, lapisan ini juga memiliki 5 unit dan menggunakan *argument return_sequences= True* untuk menghasilkan keluaran berurutan.

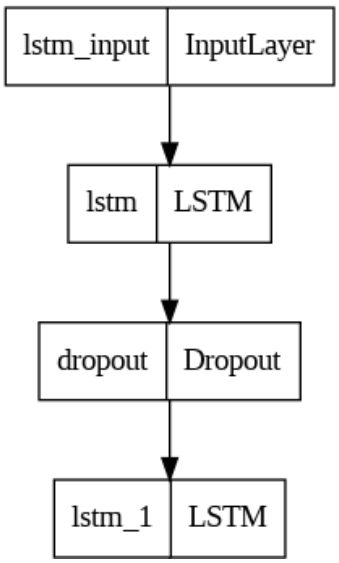
Dropout Layer 2 adalah lapisan *dropout* kedua digunakan lagi dengan tingkat *dropout* 0.2 setelah LSTM layer 2. Kemudian, LSTM *layer* 3, lapisan LSTM ketiga dalam model. Lapisan ini memiliki 5 unit, tetapi kali ini tidak menggunakan *argument return_sequences=True*, sehingga hanya menghasilkan keluaran pada *timestep* terakhir. *Dropout Layer* 3 dimana lapisan ini adalah *dropout* ketiga digunakan lagi dengan tingkat *dropout* 0.2 setelah LSTM *Layer* 3. *Dense Layer* 1, lapisan *Dense* dengan 5 unit. Lapisan Dense digunakan untuk mengubah keluaran dari *layer* LSTM menjadi bentuk yang sesuai dengan tugas yang ingin diselesaikan.

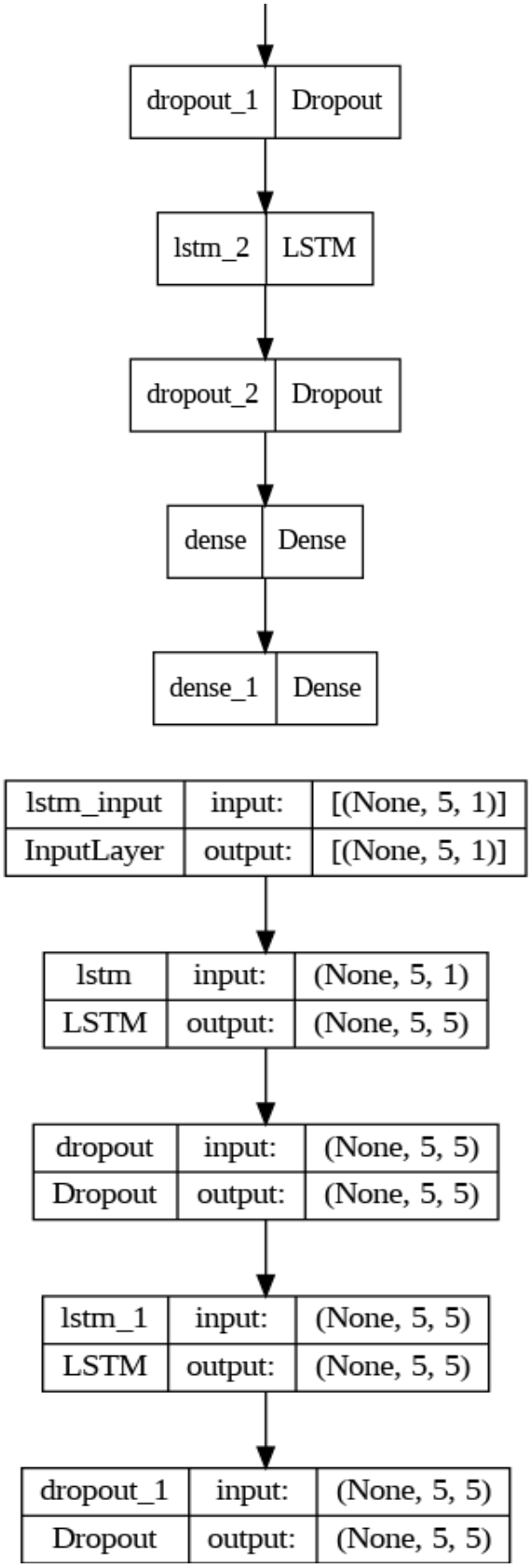
Dense layer 2 adalah lapisan *Dense* terakhir dengan 1 unit. Lapisan ini menghasilkan keluaran akhir dari model.

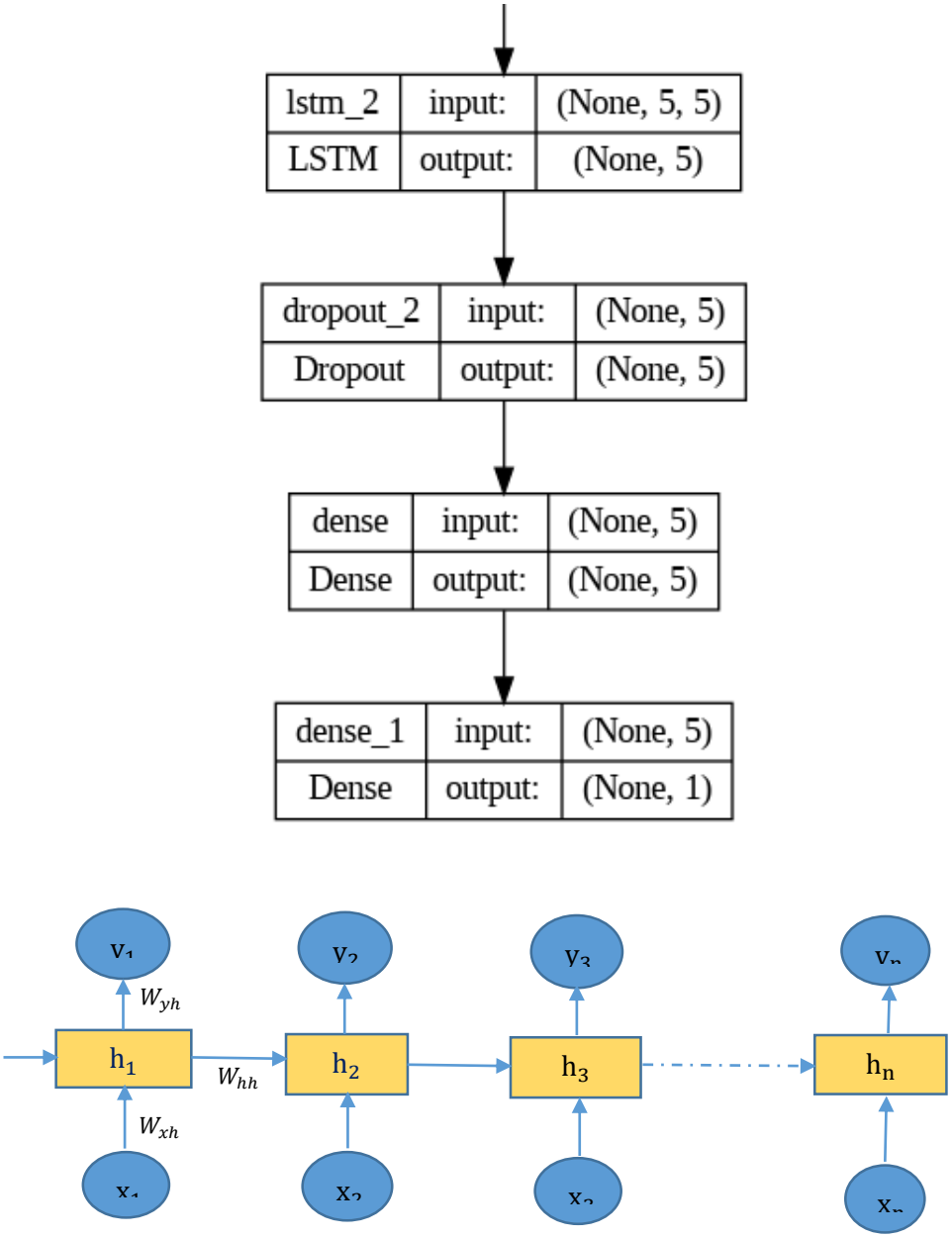
```
[42] import numpy as np
     import tensorflow as tf

     # Reshaping data agar sesuai dengan input model RNN
     X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
     X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

     # Membangun model RNN
     model = tf.keras.Sequential()
     model.add(tf.keras.layers.LSTM(units=5, return_sequences=True, input_shape=(X_train.shape[1], 1)))
     model.add(tf.keras.layers.Dropout(0.2))
     model.add(tf.keras.layers.LSTM(units=5, return_sequences=True))
     model.add(tf.keras.layers.Dropout(0.2))
     model.add(tf.keras.layers.LSTM(units=5))
     model.add(tf.keras.layers.Dropout(0.2))
     model.add(tf.keras.layers.Dense(units=5))
     model.add(tf.keras.layers.Dense(units=1))
```

Gambar 4.2. Pemanggilan Modul untuk Model *Recurrent Neural Network* (*RNN*)

Tabel 4.6. Pembuatan Model *Recurrent Neural Network* (*RNN*)




dropout_1 Dropout
lstm_2 LSTM
dropout_2 Dropout
dense Dense
dense_1 Dense
lstm_input input: [(None, 5, 1)]
InputLayer output: [(None, 5, 1)]
lstm input: (None, 5, 1)
LSTM output: (None, 5, 5)
dropout input: (None, 5, 5)
Dropout output: (None, 5, 5)
lstm_1 input: (None, 5, 5)
LSTM output: (None, 5, 5)
dropout_1 input: (None, 5, 5)
Dropout output: (None, 5, 5)



Gambar 4.3. Arsitektur *Recurrent Neural Networks* (*RNN*).

Recurrent Neural Networks (*RNN*) memiliki arsitektur yang seolah-olah berlapis, namum pada dasarnya hanya terdiri dari *neural networks* yang sederhana namun digunakan secara berulang. Setiap satu *layer neural networks* akan digunakan berulang kali pada beberapa *time step* yang berurutan. Sehingga, proses RNN ini akan menelusuri setiap lapis *neural networks* pada satu rangkaian waktu pada arsitektur RNN tersebut. Jika ingin memproses suatu pada data x yang merupakan *input layer*, notasi h adalah *hidden layer* dan y adalah prediksi atau

output , maka untuk data x_2 selanjutnya berkaitan pada data x_1 sebelumnya. Supaya punya informasi tentang data yang sebelumnya maka masukkan juga *output* dari *hidden layer* dari data sebelumnya , maka *input*nya ada 2 yaitu dari *hidden layer* yang pertama x_1 dan hidden layer kedua yaitu x_2, lalu digabungkan nilainya untuk mendapatkan *ouput* yang kedua yaitu y_2 , kemudian informasi yang dihasilkan akan digunakan pada pemprosesan data selanjutnya dan begitupun seterusnya sampai data terakhir . Pada data x_n, dimana x_n ini merupakan jumlah data. Proses RNN yang berulang-ulang ini maka dapat menyimpan hasil perhitungan pada *time step* sebelumnya untuk digunakan pada *time step* berikutnya yang merupakan kumpulan beberapa lapisan *neural networks.* Jika banyaknya *loop* yang dilakukan maka akan semakin banyak pula informasi yang perlu disimpan.

```
94/94 [==============================] - 1s 9ms/step - loss: 0.1525
Epoch 22/30
94/94 [==============================] - 1s 9ms/step - loss: 0.1509
Epoch 23/30
94/94 [==============================] - 1s 9ms/step - loss: 0.1496
Epoch 24/30
94/94 [==============================] - 1s 9ms/step - loss: 0.1504
Epoch 25/30
94/94 [==============================] - 1s 9ms/step - loss: 0.1501
Epoch 26/30
94/94 [==============================] - 1s 9ms/step - loss: 0.1497
Epoch 27/30
94/94 [==============================] - 1s 10ms/step - loss: 0.1509
Epoch 28/30
94/94 [==============================] - 1s 9ms/step - loss: 0.1496
Epoch 29/30
94/94 [==============================] - 1s 9ms/step - loss: 0.1495
```

Gambar 4.4. Hasil *Mean Squared Error* Pelatihan 30 *Epoch* (*training*)

```
[45] # Mengevaluasi model
     mse = model.evaluate(X_test, y_test, verbose=0)
     print('Mean Squared Error:', mse)

     Mean Squared Error: 0.14484049379825592
```

Gambar 4.5. Hasil Evaluasi MSE (*testing*)

Berdasarkan hasil *training* dan *testing*, didapatkan nilai yang tidak jauh berbeda. Hal ini berarti tidak terjadinya *overfitting*.

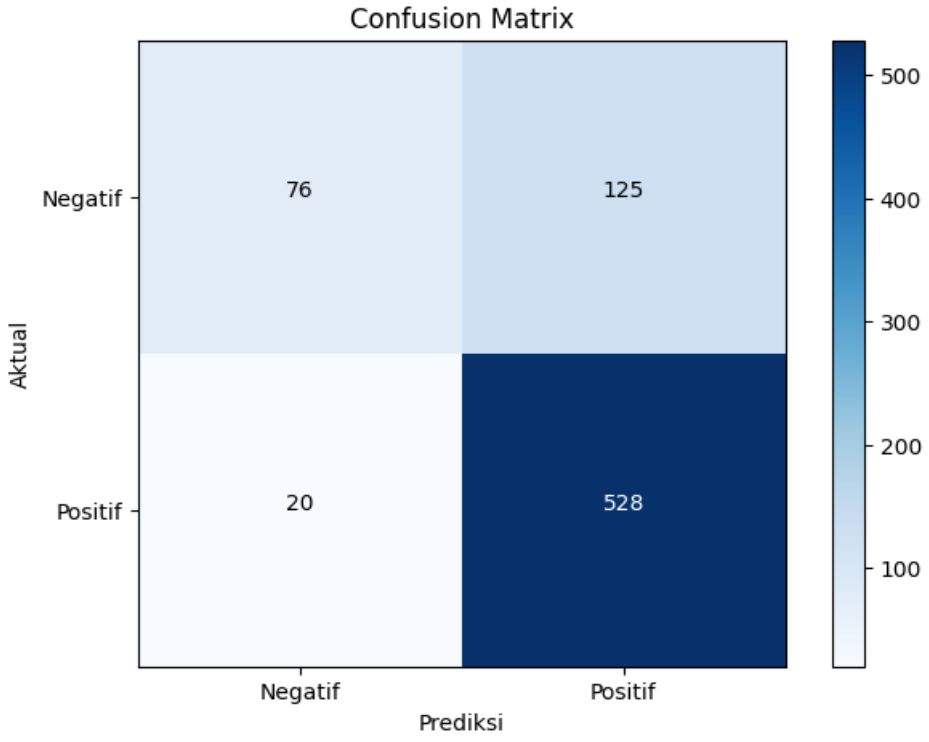
Tabel 4.7. Perbandingan Hasil *Training* dan *Testing*

Training	*Testing*
14,95 %	14,48 %

4.4.Analisa Hasil Prediksi

Pada bagian ini akan membahas dan menjelaskan mengenai hasil prediksi dari pemodelan algoritma *Recurrent Neural Network* (*RNN*) yang telah dilakukan pada proses sebelumnya.

Dalam analisis hasil prediksi ini, tahap evaluasi model dilakukan menggunakan *Confusion Matrix* yang dapat memberikan perbandingan hasil, telah dilakukan menggunakan model RNN denga hasil klasifikasi sebenarnya yang dapat dilihat pada nilai *accuracy, precision dan recall* seperti yang ditunjukkan dalam Gambar 4.6 dibawah ini



Gambar 4.6. *Confusion Matrix* RNN

Berdasarkan gambar di atas, kita dapat mengetahui bahwa hasil *Confusion Matrix* RNN menunjukkan nilai *true positive* (hujan) 528, *true negative* (tidak hujan) 76, *false positive* (hujan) 125 dan *false negative* (tidak hujan) 20. Untuk melakukan analisa hasil prediksi model RNN, digunakannya perhitungan *accuracy*, *precision,* dan *recall* dengan menggunakan rumus berikut ini :

$$Accuracy = \frac{528 + 76}{528 + 76 + 125 + 20} \times 100 = 80.64\%$$

$$Precision = \frac{528}{528 + 125} \times 100 = 80.76\%$$

$$Recall = \frac{528}{528 + 20} \times 100 = 96.35\%$$

Berdasarkan hasil analisa menggunakan *confusion matrix* telah didapatkan nilai *accuracy*, *precision,* dan *recall* dari data prediksi curah hujan di Ogan Komering Ilir, yakni nilai *accuracy* sebesar 80,64% yang artinya ke akuratan prediksi tersebut terbilang baik atau aktual. Selanjutnya nilai *precision* sebesar 80,76% hasil ini menunjukkan bahwa data yang bernilai positif dan nilai *recall* sebesar 96,35% hasil ini menunjukkan bahwa data yang terklasifikasikan dengan benar oleh model adalah positif.