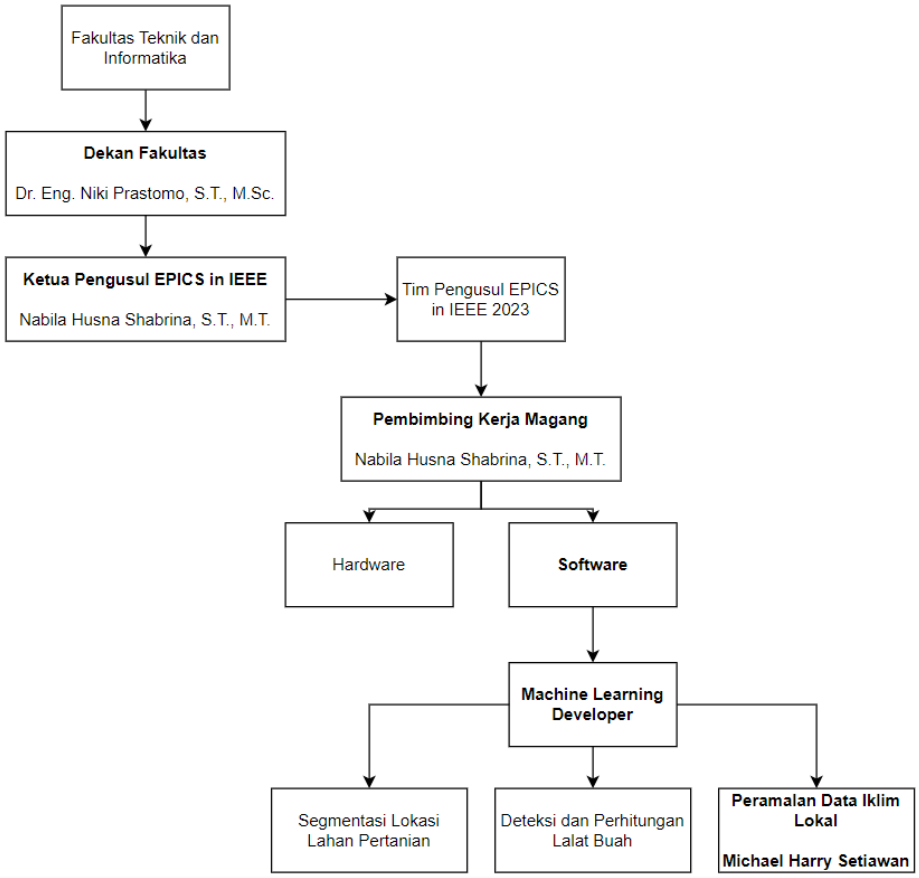
# BAB III
# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Koordinasi

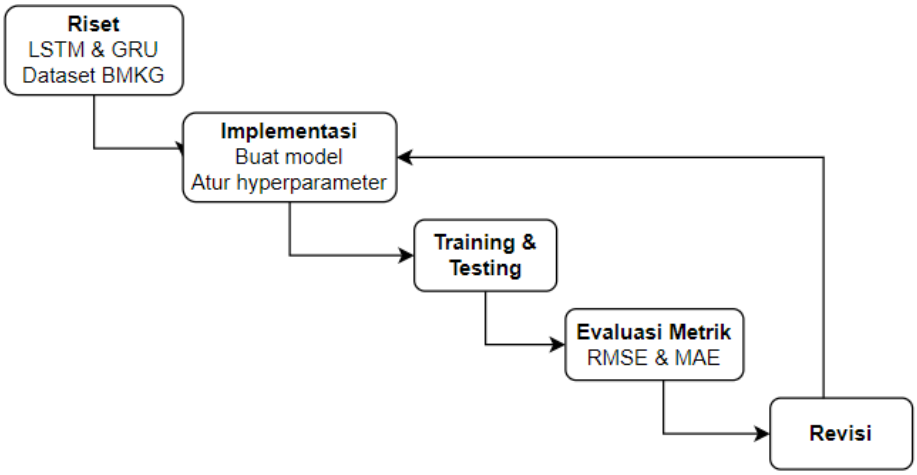



Gambar 3.1 Kedudukan posisi pemangang

Posisi pemagang disini sebagai *machine learning engineer* dapat dilihat pada Gambar 3.1, yang secara khusus difokuskan pada pengembangan model kecerdasan buatan untuk melakukan peramalan terhadap kondisi iklim lokal. Dalam magang ini data yang harus diprediksi adalah data iklim seperti temperatur, kelembapan, curah hujan dan intensitas cahaya, karena parameter tersebut yang menjadi penanda kemunculan hama.

Proses magang dimulai dengan tahap riset yang menyeluruh terkait dengan aspek-aspek yang relevan dengan pembuatan model, termasuk pencarian *dataset* sekunder, penyesuaian *hyperparameter*, pemahaman algoritma *sliding window*,

pemahaman dan pemilihan metrik pengujian, serta *layers* dalam model dengan fokus khusus pada riset terkait *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU). Setelah tahap riset selesai, langkah berikutnya adalah implementasi dari pembuatan model sesuai dengan apa yang sudah dicari, dan melakukan proses *training* serta *testing*. Hasil dari proses *testing* kemudian akan dievaluasi dengan metrik pengujian yang sudah dipilih untuk menilai kualitas dari model. Jika hasil evaluasi menunjukkan kinerja model yang belum memuaskan, langkah selanjutnya adalah melakukan revisi pada model untuk meningkatkan kualitas prediksi, ilustrasi dapat dilihat pada gambar 3.2.




Gambar 3.2 Workflow

## 3.2 Tugas dan Uraian Kerja Magang

### *3.2.1 Timeline*

Berikut uraian kegiatan yang dilaksanakan selama magang semester 6:

Tabel 3.1 Timeline kegiatan

| Bulan | Minggu | Kegiatan |
|---|---|---|
| Februari | 1 | Riset untuk mencari informasi mengenai pengolahan data, fungsi-fungsi, serta acuan untuk melakukan prediksi *time series*. |

| | | |
|---|---|---|
| | 2 | Fokus ditujukan pada pencarian *dataset* sekunder untuk melakukan *training* model sementara dari berbagai sumber, seperti Kaggle dan BMKG. |
| | 3 | Mencari model yang sudah pernah dibuat orang lain untuk menjadi contoh utama. |
| | 4 | Survey lokasi di Sleman, Yogyakarta. |
| Maret | 1 | Membuat *pre-processing* untuk menghilangkan data yang tidak diinginkan seperti menghilangkan data *null*, data bernilai 0, dan data yang nilainya terlalu besar. |
| | 2 | Membuat dan merancang model yang relevan dengan data sekunder yang dimiliki. |
| | 3- 4 | Evaluasi model yang sudah dibuat, terutama melihat metrik pengujian untuk kembali melakukan revisi agar kualitas model kecerdasan buatan bisa ditingkatkan lebih lanjut. |
| April | 1-3 | Tetap menggunakan data sekunder yang sudah tersedia untuk proses pelatihan, serta evaluasi model untuk peningkatan kualitas model. |
| | 4 | Persiapan untuk *save* model yang dilatih menggunakan data sekunder sebagai *backup*, dan nantinya pelatihan selanjutnya akan dilakukan dengan data primer yang sudah didapatkan. |
| Mei | 1-4 | Periode pembuatan laporan magang. |

### 3.2.2 Pencarian dan Pembagian Dataset

Dilakukan pencarian *dataset* sekunder sebagai langkah uji coba terhadap model sebelum memanfaatkan *dataset* primer. Penting untuk memastikan bahwa *dataset* sebisa mungkin menyerupai data primer yang akan digunakan nantinya. Hal ini bertujuan agar hasil pelatihan model pada kedua *dataset* tidak terlalu berbeda. *Dataset* yang diperoleh berupa data iklim, termasuk temperatur rata-rata, kelembapan relatif, dan curah hujan di daerah Sleman. Data tersebut diambil dari sumber Badan Meteorologi, Klimatologi, dan Geofisika (BMKG) dengan rentang

waktu tiga tahun (https://dataonline.bmkg.go.id/home). Serta data intensitas cahaya yang diambil dari *Eastern Ecological Science Center* di Amerika (https://www.sciencebase.gov/catalog/item/5f18545682cef313ed8431a9). Data tersebut digunakan sesuai dengan informasi yang dijabarkan oleh [1], bahwa data faktor cuaca untuk memprediksi lalat buah adalah curah hujan, kelembapan udara, suhu udara, dan kecepatan angin. Tetapi setelah melakukan diskusi, kecepatan angin diganti menjadi intensitas cahaya.

- **Data Temperatur Rata-rata**

  Kalkulasi dari temperatur rata-rata dilakukan dengan perhitungan:

  $$(2xSuhu\ jam\ 07.00\ +\ Suhu\ jam\ 13.00\ +\ Suhu\ jam\ 18.00)/4 \quad (1)$$

- **Data Kelembapan Relatif**

  Kalkulasi dari kelembapan relatif dilakukan dengan perhitungan:

  $$(2xRH\ jam\ 07.00\ +\ RH\ jam\ 13.00\ +\ RH\ jam\ 18.00)/4 \quad (2)$$

- **Data Curah Hujan**

  Untuk data curah hujan sendiri diamati tiap jam 07.00.

- **Data Intensitas Cahaya**

  Data intensitas cahaya diamati setiap jam mulai dari jam 00.00 hingga jam 23.00, menghasilkan total 24 data dalam satu hari. Namun, kebutuhan akan nilai intensitas cahaya adalah per hari. Oleh karena itu, nilai intensitas cahaya pada periode malam hari mulai dari jam 19.00 hingga jam 07.00 dihapus, sedangkan nilai intensitas cahaya pada periode siang hari diambil rata-ratanya. Proses ini penting untuk menggambarkan kondisi yang lebih relevan, karena kebutuhan cahaya umumnya terjadi pada siang hari. Berikut contoh data iklim dari BMKG serta dari *Eastern Ecological Science Center*:

Tabel 3.2 *Dataset* temperatur rata-rata, kelembapan relatif, dan curah hujan

| Tanggal | Tavg | RH_avg | RR |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1/1/2021 | 25.8 | 90.0 | 18.8 |
| 2/1/2021 | 26.4 | 85.0 | 5.8 |
| 3/1/2021 | 25.8 | 85.0 | 0.0 |
| 4/1/2021 | 26.1 | 83.0 | 0.0 |
| 5/1/2021 | 26.3 | 86.0 | 0.0 |

Tabel 3.3 *Dataset* intensitas cahaya

| Date | VO2_1U |
|---|---|
| 2015-10-01 | 4304.585623 |
| 2015-10-02 | 3567.747029 |
| 2015-10-03 | 12694.238332 |
| 2015-10-04 | 8571.660649 |
| 2015-10-05 | 24897.752748 |

Data kemudian dibagi menjadi data *training* dan data *testing* seperti berikut:

Tabel 3.4 Pembagian *dataset*

| Data | Total data | *Training* | *Testing* |
|---|---|---|---|
| Kelembapan Relatif | 1096 | 879 | 213 |
| Rata-rata Temperatur | 1096 | 880 | 213 |
| Precipitation (mm/d) | 1096 | 876 | 210 |
| Light Intensity | 1461 | 1247 | 214 |

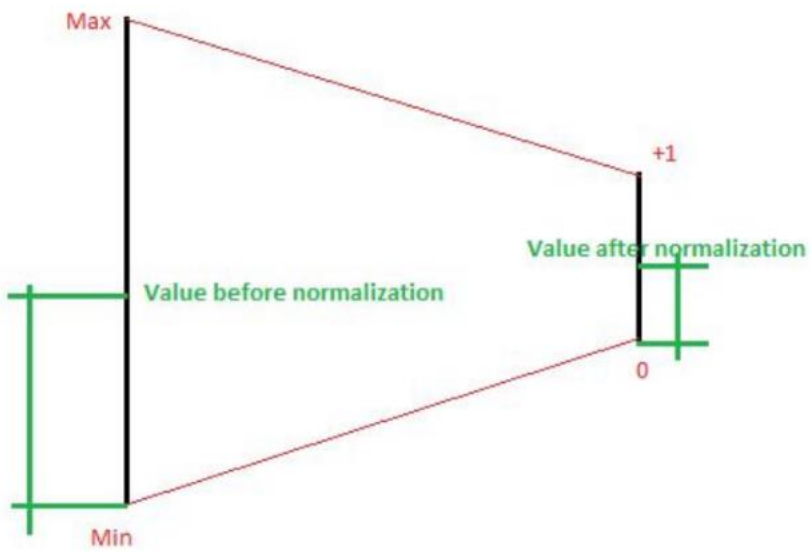
Pada data *training* dan *testing*, terdapat tiga macam dimensi pada bentuk data, yaitu (x, 1, 1), kecuali untuk data curah hujan (precipitation) yang memiliki bentuk (x, 4, 1). Dimensi pertama mengindikasikan jumlah data yang digunakan, sehingga merupakan komponen yang paling penting karena mencerminkan total data yang terlibat dalam proses pelatihan. Dimensi kedua merepresentasikan *sliding window*, contohnya dari sebaran data 1, 2, 3, 4, 5 dan menggunakan *sliding window* dengan

ukuran 4, maka akan menghasilkan *array* [1, 2, 3, 4], [2, 3, 4, 5]. Penggunaan algoritma *sliding window* menjadi salah satu cara untuk model mengenali data lebih baik dengan cara memperkenalkan *overlap* dari data sebelumnya sehingga pola dapat lebih mudah dikenal oleh model *machine learning*. Dimensi ketiga menjelaskan bahwa setiap elemen dalam *array* tersebut mengacu pada satu data tunggal.

### *3.2.3 Feature Scaling*

*Feature scaling* merupakan sebuah metode untuk menyetarakan skala dari beberapa fitur [2]. Dalam magang ini ada 4 fitur yang digunakan, seperti kelembapan relatif, temperatur rata-rata, curah hujan, dan intensitas cahaya. Maka dari itu penting untuk menyetarakan skala dari masing-masing parameter yang digunakan. Ada 2 macam feature scaling yang umum dikenal, yaitu normalisasi dan standarisasi atau normalisasi *z-score*.

Normalisasi data merupakan proses untuk mengubah skala data ke dalam *range* tertentu, misalnya 0 sampai 1. Normalisasi data penting dilakukan ketika terdapat perbedaan skala yang besar pada fitur yang berbeda [3]. Bisa dilihat pada Gambar 3.3.



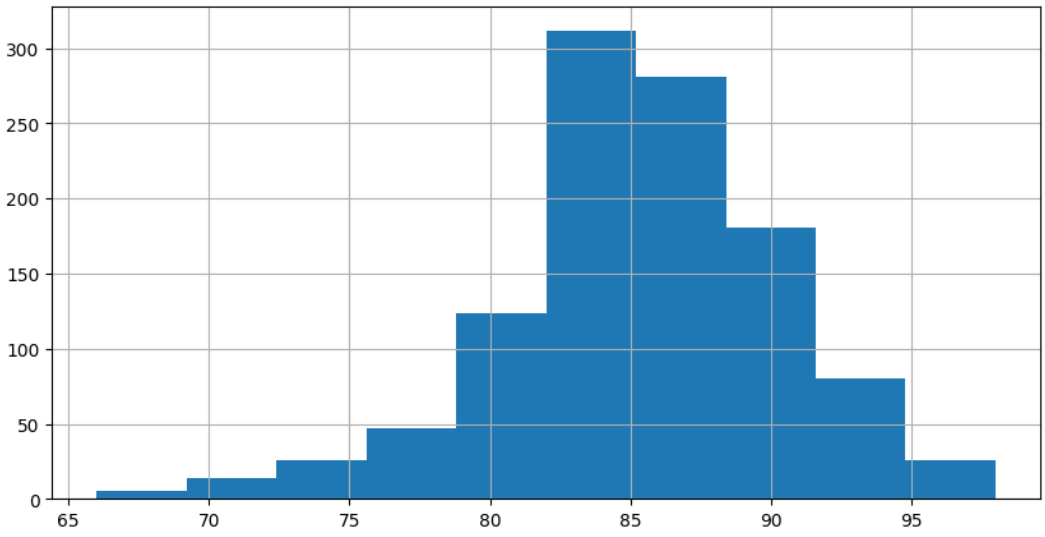

Gambar 3.3 Normalisasi data [3]

Standarisasi atau normalisasi *z-score*, yang membuat data dinormalisasikan dengan pengaruh rata-rata dan standar deviasinya. Normalisasi dan standarisasi dapat dilihat dengan persamaan berikut:

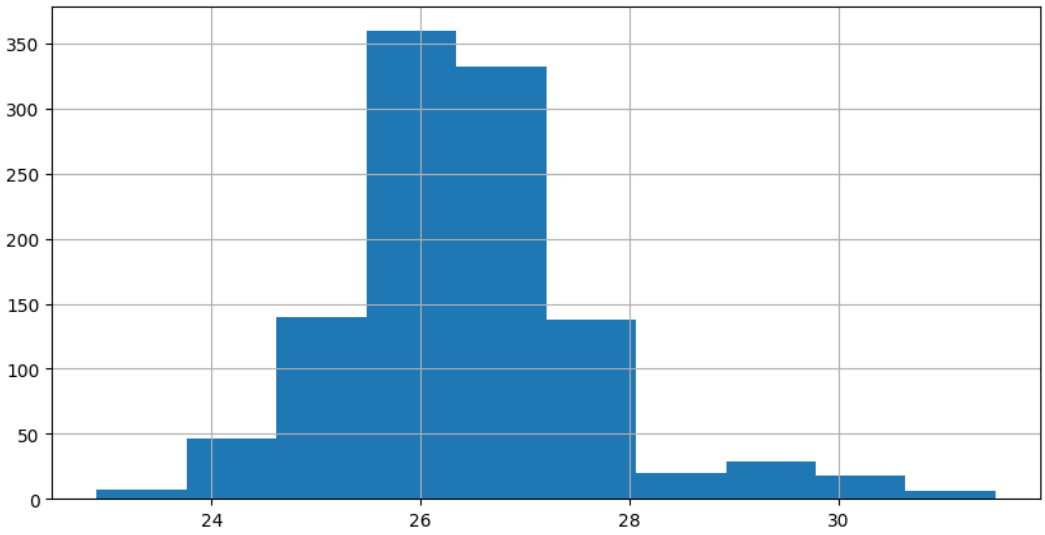
$$Normalisasi \rightarrow x' = \frac{x-min}{max-min} \quad (3)$$

$$Standarisasi \rightarrow x' = \frac{x-mean}{sd} \quad (4)$$

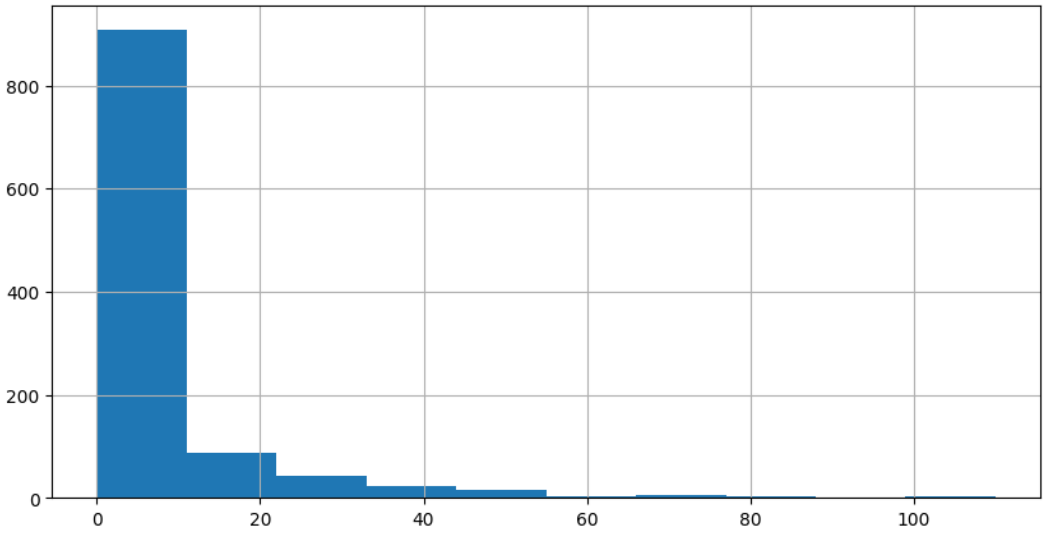
Menurut [4], normalisasi *z-score* dikonklusikan sebagai normalisasi yang terbaik untuk segala domain, permasalahan, algoritma atau data. Tetapi, normalisasi *z-score* hanya bisa digunakan ketika data mengikuti sebaran normal (*gaussian distribution*) [3].



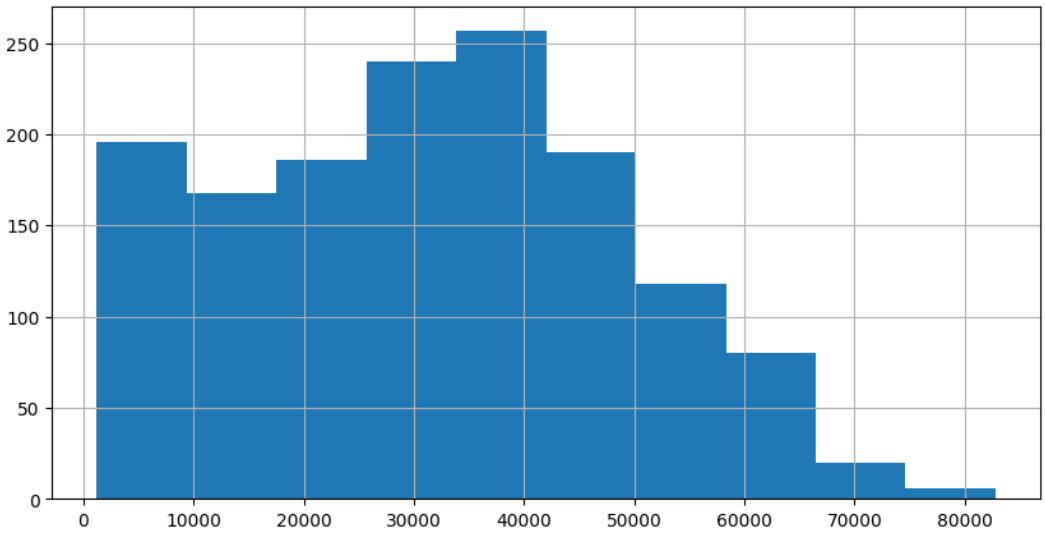

Gambar 3.4 Sebaran data kelembapan relatif




Gambar 3.5 Sebaran data temperatur rata-rata




Gambar 3.6 Sebaran data curah hujan




Gambar 3.7 Sebaran data intensitas cahaya

Tabel 3.5 Hasil uji shapiro-wilk

| Parameter uji | p-value |
|---|---|
| Kelembapan | 2.481e-10 |
| Temperatur | 1.774e-19 |
| Curah hujan | 0.0 |
| Intensitas cahaya | 7.126e-14 |

Dari Gambar 3.4, Gambar 3.5, Gambar 3.6, dan Gambar 3.7, serta dari Tabel 3.5. Disimpulkan bahwa data kelembapan relatif, temperatur rata-rata, curah hujan dan intensitas cahaya semuanya mengikuti sebaran data yang tidak normal. Sehingga normalisasi *z-score* tidak dapat digunakan. Menurut [5], model LSTM bekerja lebih baik pada skala 0 sampai 1. Sehingga normalisasi diaplikasi dengan batas 0 sampai 1.

### 3.2.4 LSTM dan GRU

Ada dua model kecerdasan buatan yang akan diuji, yaitu model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU). *Layer* model LSTM yang digunakan adalah sebagai berikut:

Tabel 3.6 *Layer* model LSTM

| *Layer* | *Output Shape* | Parameter |
|---|---|---|
| LSTM | (*None*, 1, 128) | 66560 |
| *Dropout* | (*None*, 1, 128) | 0 |
| *BatchNormalization* | (*None*, 1, 128) | 512 |
| LSTM | (*None*, 1, 128) | 131584 |
| *Dropout* | (*None*, 1, 128) | 0 |
| *BatchNormalization* | (*None*, 1, 128) | 512 |
| LSTM | (*None*, 128) | 131584 |
| *Dropout* | *(None*, 128*)* | 0 |
| *BatchNormalization* | (*None*, 128) | 512 |
| *Dense* | (*None*, 1) | 129 |

Berikut juga *layer* pada model GRU yang digunakan:

Tabel 3.7 *Layer* model GRU

| *Layer* | *Output Shape* | Parameter |
|---|---|---|
| GRU | (*None*, 1, 128) | 50304 |
| *Dropout* | (*None*, 1, 128) | 0 |
| *BatchNormalization* | (*None*, 1, 128) | 512 |
| GRU | (*None*, 1, 128) | 99072 |
| *Dropout* | (*None*, 1, 128) | 0 |
| *BatchNormalization* | (*None*, 1, 128) | 512 |
| GRU | (*None*, 128) | 99072 |
| *Dropout* | (*None*, 128) | 0 |
| *BatchNormalization* | (*None*, 128) | 512 |
| *Dense* | (*None*, 1) | 129 |

*Output shape* merupakan representasi *array* yang dihasilkan dari suatu *layer* dalam model. Dimensi pertama dari *output shape* adalah *batch size*, yang sering

kali direpresentasikan dengan nilai *None* karena tidak dituliskan secara eksplisit pada pembuatan model. Terdapat *layer* yang memiliki dimensi 2D dan 3D. Perbedaan utama antara keduanya terletak pada parameter "*return sequence*" yang menentukan apakah *output* dari *layer* tersebut akan dijadikan *input* untuk *layer* selanjutnya atau tidak. Misalnya, jika parameter "*return sequence*" diatur sebagai *True*, *output* dari *layer* tersebut berupa urutan data yang akan diteruskan ke *layer* selanjutnya, sehingga menjadikan *output* tersebut berdimensi 3D. Angka 128 yang disebutkan merupakan jumlah *unit* atau *neuron* yang terdapat pada *layer* tersebut, yang mempengaruhi kompleksitas dan kapasitas model.

*Parameter* menjadi representasi *neural network* yang sudah *fully connected layer* artinya semua *node* pada *neural network* sudah terhubung. Jumlah parameter ini didapatkan dengan persamaan berikut:

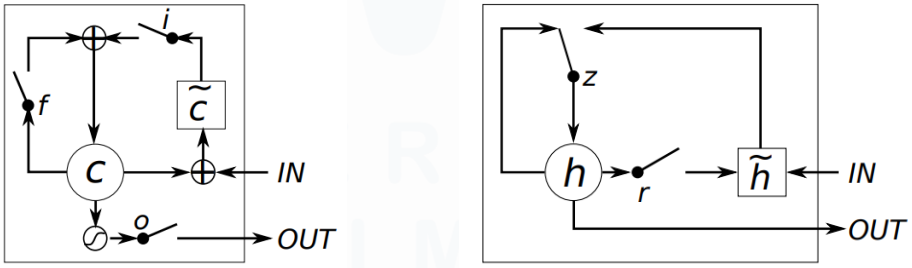
$$LSTM = 4((input\ size + unit\ size) * unit\ size + unit\ size) \quad (5)$$

$$LSTM = 4((1 + 128) * 128 + 128) = 66560 \quad (6)$$

$$GRU = 3\big((input\ size + unit\ size) * unit\ size + (2 * unit\ size)\big) \quad (7)$$

$$GRU = 3((1 + 128) * 128 + (2 * 128)) = 50304 \quad (8)$$

Dari Gambar 3.8 i, f, o masing-masing berarti *input*, *forget* dan *output*. c merupakan sel memori dan c` merupakan sel memori baru. Pada diagram GRU z, r masing-masing merupakan *update* dan *reset*. h merupakan fungsi aktivasi dan h` merupakan kandidat aktivasi.




(a) Long Short-Term Memory (b) Gated Recurrent Unit

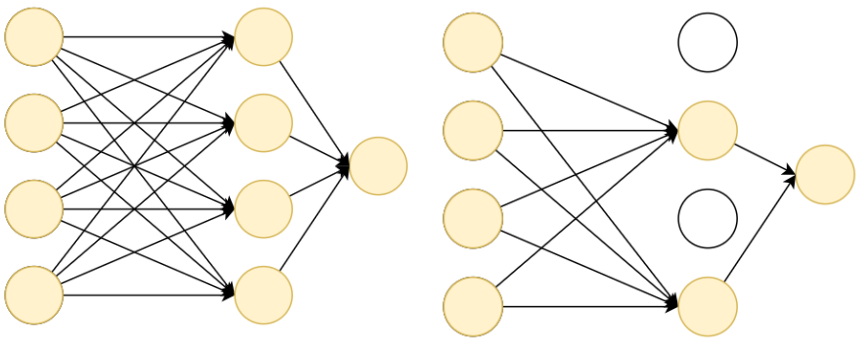
Gambar 3.8 *Block diagram* LSTM dan GRU. [6]

Pada LSTM, ketika terdapat *input* baru, terjadi proses penjumlahan langsung antara *input* dan sel memori. Hal ini memungkinkan LSTM untuk mengingat informasi yang diberikan dari *input* sebelumnya, selain itu LSTM memiliki *forget gate*, dengan *gate* ini LSTM dapat mengontrol dan mengelola memorinya dengan lebih baik, sehingga informasi yang dianggap kurang relevan dapat dihilangkan atau dilupakan.

Sementara itu, pendekatan pada model GRU jauh lebih sederhana. GRU menggunakan *update gate* z dan kandidat aktivasi h. *Update gate* z mengatur seberapa banyak informasi dari *unit* sebelumnya yang akan diteruskan. Sedangkan kandidat aktivasi h adalah nilai yang dihasilkan oleh unit tersebut berdasarkan input dan informasi dari unit sebelumnya [6].

*Deep learning* sering dipilih meskipun jumlah data pelatihan awalnya terbatas. Hal ini karena seiring berjalannya waktu, jumlah data akan terus bertambah, yang pada akhirnya meningkatkan kompleksitas data. *Deep learning* memiliki kemampuan untuk mengekstrak pola yang rumit dari data yang besar, sehingga menjadi alat yang efektif dalam analisis *Big Data*. Di sisi lain, teknik *machine learning* konvensional cenderung kurang efisien dalam mengekstrak pola yang kompleks dan tidak linear dari *Big Data* [7].

### 3.2.5 Dropout



(a) Tanpa *dropout* (b) Dengan *dropout*

Gambar 3.9 Ilustrasi *dropout*.

Teknik regularisasi adalah strategi untuk mengurangi kesalahan saat melatih model [8]. Hal ini khususnya penting dalam jaringan *neural network* dengan banyak *hidden layer*, di mana peningkatan jumlah *layer* dapat meningkatkan risiko *overfitting*. *Overfitting* membuat model menjadi terlalu kaku dan kurang mampu mengenali pola pada data baru yang tidak terlihat selama pelatihan.

Untuk mengatasi masalah ini, digunakanlah teknik *layer dropout*. *Layer dropout* mematikan sebagian kecil dari *neuron* di *hidden layer* secara acak selama proses pelatihan. Dengan cara ini, *layer dropout* memaksa model untuk mempelajari fitur-fitur yang lebih umum dan mencegah potensi *overfitting*. Hal ini meningkatkan fleksibilitas model dalam mengenali pola pada data yang tidak terlihat sebelumnya. Ilustrasi dari penggunaan *layer dropout* dapat dilihat pada Gambar 3.9, dalam model yang menggunakan *layer dropout*, beberapa unit dalam *hidden layer* dimatikan secara acak selama proses pelatihan.

### 3.2.6 Batch Normalization

Salah satu permasalahan yang sering muncul dalam proses pelatihan model *neural network* adalah "*internal covariate shift*", yaitu perubahan yang terjadi pada distribusi data di dalam jaringan *neural network* selama proses pelatihan [9]. Permasalahan ini akan mempersulit proses pelatihan model. Serupa halnya dengan proses pembelajaran pada manusia, ketika seseorang sedang menjelaskan sesuatu, pemahaman tiap orang akan berbeda, maka dari itu harus dilakukan sebuah normalisasi agar pemahaman tiap orang tidak berbeda.

### 3.2.7 Metrik Pengujian

Model kemudian diuji dengan beberapa metrik untuk mengetahui kualitasnya, beberapa metrik digunakan untuk menilai model dengan lebih baik sesuai dengan yang dikatakan [10], bahwa kombinasi dari beberapa metrik terkadang diperlukan untuk menilai performa model. Metrik yang digunakan antara lain, *Mean Absolute Error* (MAE) dan *Root Mean Square Error* (RMSE) masing-masing dengan rumus sebagai berikut:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1}(yi-\hat{y}i)^2}{N}} \quad (9)$$

$$MAE = \frac{\sum_{i=0}^{N-1}|yi-\hat{y}i|}{N} \quad (10)$$

N merupakan jumlah sampel dalam dataset, yi merupakan nilai sebenarnya dari sampel ke-i dan ŷi merupakan nilai prediksi dari sampel ke-i. Namun, RMSE sering menjadi metrik utama dalam pengujian model, terutama dalam konteks meteorologi, kualitas udara, dan penelitian iklim, seperti yang disebutkan dalam referensi [10], bahwa RMSE sering dianggap sebagai standar metrik statistikal untuk mengukur tingkat kesalahan dari model prediksi.

### 3.2.8 Sintaks Kode

```
scaler = MinMaxScaler(feature_range= (0, 1))
scaled_data = scaler.fit_transform(np.array(dataset))
```

Gambar 3.10 Sintaks normalisasi data

Gambar 3.10 menunjukan pengaplikasian normalisasi data yang digunakan yaitu dengan skala normalisasi 0 sampai 1.

```
def lstm_model():
    model = Sequential()
    model.add(LSTM(units = 128, return_sequences=True, input_shape=(X_train.shape[1], 1)))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(LSTM(units = 128, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(LSTM(units = 128, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam')
    model.summary()

    return model
```

Gambar 3.11 Model LSTM

```
def gru_model():
    model = Sequential()
    model.add(GRU(units = 128, return_sequences=True, input_shape=(X_train.shape[1], 1)))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(GRU(units = 128, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(GRU(units = 128, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam')
    model.summary()

    return model
```

Gambar 3.12 Model GRU

Gambar 3.11 dan Gambar 3.12 menunjukan *layer* yang digunakan dalam model LSTM dan GRU. *Sequential* menunjukkan bahwa kategori model *machine learning* adalah *feedforward neural networks*, artinya tidak ada feedback yang diberikan oleh *output* terhadap *input* [11]. Penggunaan *layer* LSTM/GRU, *dropout*, dan *batch normalization* disini didapatkan dengan eksperimentasi yang dilakukan berulang kali, model ini untuk sekarang memperoleh hasil metrik yang paling optimal. *Layer dense* memiliki satu *unit neuron*, menunjukkan bahwa *layer* ini menjadi output dari model. Penggunaan *activation function linear* dikarenakan model ditujukan untuk memprediksi data *time-series* sehingga diperlukan model regresi, dan *activation function linear* ini membuat model berfungsi sebagai model regresi linear [12].

```
history_lstm = lstm.fit(X_train, y_train,
                        epochs=250, batch_size=64,
                        validation_split=0.2,
                        verbose=2)
```

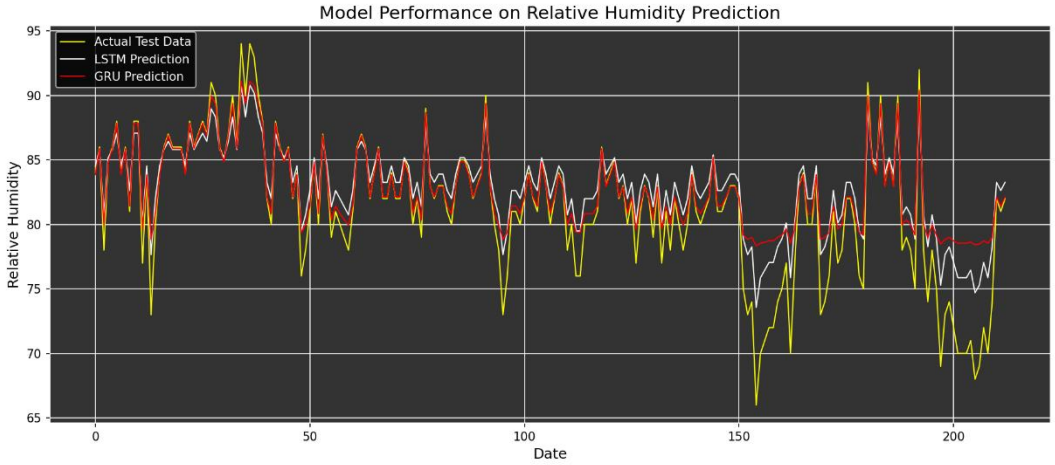
Gambar 3.13 *Hyperparameter* pelatihan LSTM

```
history_gru= gru.fit(X_train, y_train,
                     epochs=250, batch_size=64,
                     validation_split=0.2,
                     verbose=2)
```

Gambar 3.14 *Hyperparameter* pelatihan GRU

Gambar 3.13 dan gambar 3.14 menunjukan konfigurasi hyper parameter yang digunakan pada proses pelatihan. *Epoch* pada *hyperparameter* berarti siklus pelatihan, 250 *epoch* berarti 250 siklus pelatihan. *Batch size* berarti jumlah data yang dibutuhkan untuk *update* model. *Validation split* berfungsi untuk menyisihkan sebagian dari data pelatihan untuk validasi. Contohnya dengan data pelatihan sebanyak 879, sebanyak 176 data akan digunakan untuk validasi dan 703 data untuk pelatihan. Dengan *batch size* 64 berarti data pelatihan akan dibagi menjadi 11 *batch* masing-masing dengan 64 data. Artinya 1 *epoch* akan melewati 11 *batch* dan akan mengalami *update* pada model sebanyak 11 kali, dengan total 250 *epoch* berarti model akan mengalami *update* sebanyak 2750 kali.

### 3.2.9 Hasil dan Pembahasan




Gambar 3.15 Hasil prediksi kelembapan relatif

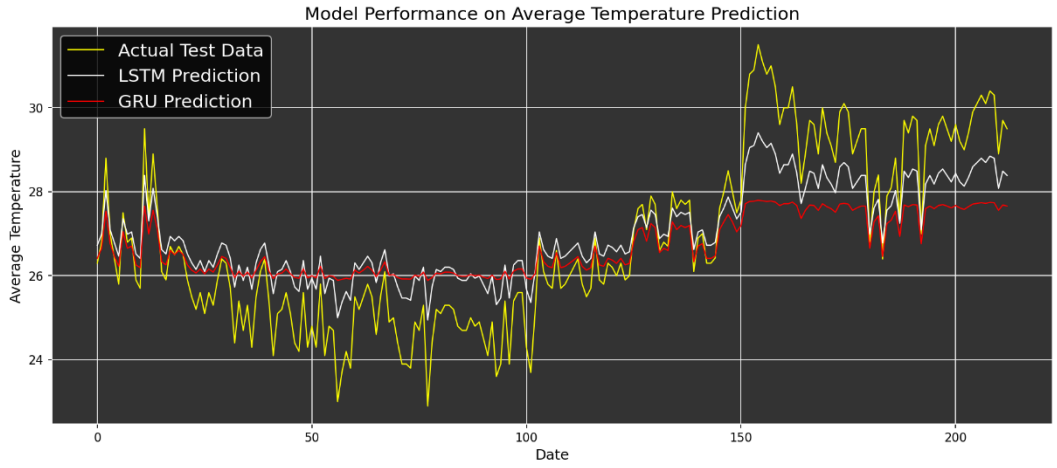
Tabel 3.8 *Output* metrik kelembapan relatif

| Metrik | LSTM | GRU |
|---|---|---|
| RMSE | 4.17065 | 4.44829 |
| MAE | 3.15413 | 3.36938 |
| MAX | 13.12603 | 12.98658 |

Dilihat dari Gambar 3.15 dan melihat dari Tabel 3.8, kedua model mampu memprediksi hasil dengan tingkat kesamaan yang cukup baik. Namun, observasi

menunjukkan bahwa LSTM lebih unggul dalam menangkap kompleksitas data. Contohnya, pada sumbu x sekitar data 150 dan 200, ketika kelembapan mengalami penurunan signifikan, LSTM mampu mengenali pola tersebut meskipun dengan tingkat akurasi yang cukup rendah. Di sisi lain, GRU tidak mampu mengenali pola tersebut dengan baik atau bahkan sama sekali.

Hal ini menunjukkan bahwa LSTM memiliki kemampuan yang lebih baik dalam menangkap dan memodelkan pola-pola yang kompleks pada data kelembapan, bahkan pada situasi ketika terjadi fluktuasi data seperti pada contoh Gambar 3.5.




Gambar 3.16 Hasil prediksi temperatur rata-rata

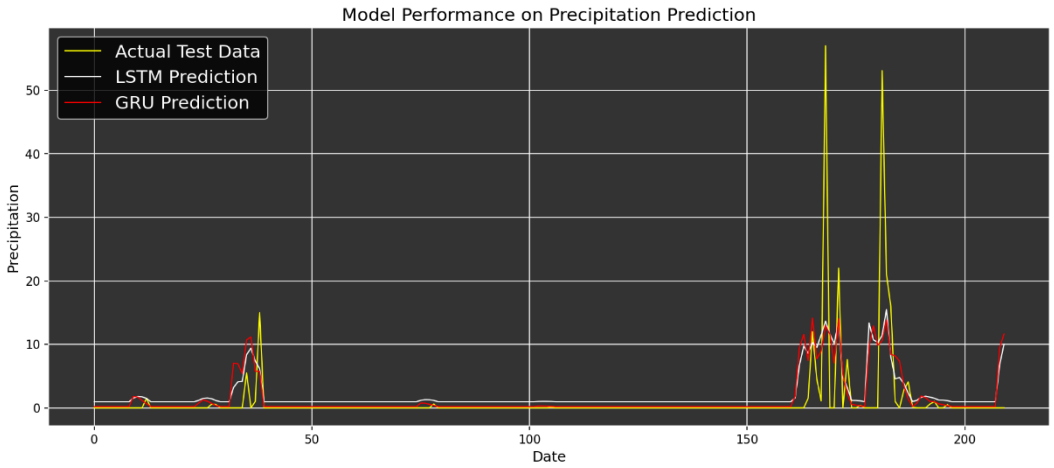
Tabel 3.9 *Output* metrik temperatur rata-rata

| Metrik | LSTM | GRU |
|---|---|---|
| RMSE | 1.19045 | 1.47135 |
| MAE | 1.00406 | 1.20670 |
| MAX | 3.30428 | 3.72492 |

Dalam pengujian temperatur, terlihat bahwa data memiliki distribusi yang cukup besar, dengan rentang temperatur dari sekitar 23 hingga hampir 32 derajat Celsius. Meskipun demikian, prediksi yang dilakukan oleh model LSTM maupun GRU cenderung berada di tengah-tengah rentang tersebut, kira-kira antara 25

hingga 29 derajat Celsius, seperti yang terlihat dalam Gambar 3.16. Serta bisa dilihat juga dari Tabel 3.9, metrik model LSTM memberikan hasil yang lebih baik.

Data yang kompleks dan tidak stasioner seperti ini menjadi salah satu tantangan dalam pengembangan model, terutama dengan jumlah data yang sedikit model tidak bisa mempelari pola tersebut dengan baik. Dapat dilihat bahwa model LSTM cenderung memberikan hasil prediksi yang lebih baik daripada model GRU, seperti yang tercermin dari metrik evaluasi yang digunakan. Hal ini menunjukkan bahwa LSTM mampu menangkap pola-pola yang lebih kompleks dalam data temperatur, bahkan dalam situasi di mana data tidak stasioner dan memiliki fluktuasi yang signifikan.




Gambar 3.17 Hasil prediksi curah hujan

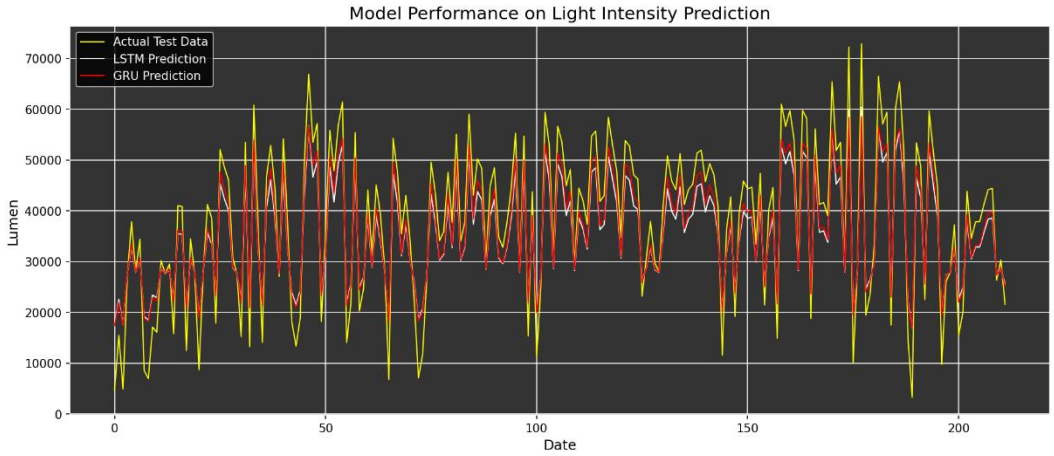
Tabel 3.10 *Output* metrik presipitasi

| Metrik | LSTM | GRU |
|---|---|---|
| RMSE | 5.93998 | 6.08887 |
| MAE | 2.44163 | 1.97807 |
| MAX | 52.14475 | 52.90686 |

Curah hujan menjadi salah satu parameter iklim yang cukup sulit untuk dilatih, salah satu permasalahannya adalah mayoritas dari data bernilai 0. Nilai 0 tersebut menandakan hari tanpa hujan. Meskipun demikian, data dengan nilai 0 ini tidak

dapat dihapus karena memiliki signifikansi penting dalam analisis curah hujan. Selain itu, fluktuasi data curah hujan juga bisa sangat besar, di mana nilai yang awalnya 0 bisa berubah drastis ke nilai yang lebih tinggi, seperti yang terlihat dalam Gambar 3.17.

Dalam situasi seperti ini, model LSTM dan GRU mungkin memberikan hasil prediksi yang serupa, mengingat kompleksitas dan fluktuasi data yang tinggi dengan jumlah data yang sedikit. Dari Tabel 3.10, nilai RMSE dari model LSTM sedikit lebih baik daripada model GRU, namun nilai MAE model LSTM signifikan lebih besar. Sehingga tidak bisa disimpulkan model mana yang lebih baik.




Gambar 3.18 Hasil prediksi intensitas cahaya

Tabel 3.11 *Output* metrik intensitas cahaya

| Metrik | LSTM | GRU |
|---|---|---|
| RMSE | 17029.10456 | 17252.04821 |
| MAE | 13070.87016 | 13160.19174 |
| MAX | 49954.53527 | 48274.45714 |

Melihat dari Gambar 3.18, hasil prediksi secara visual cukup baik. Namun, yang menjadi pertanyaan adalah mengapa nilai metrik evaluasi menjadi sangat besar. Hal ini bisa disebabkan oleh nilai data yang relatif besar dalam dataset. Misalnya,

rentang nilai lumen rata-rata harian bisa mencapai 70000. Dari Tabel 3.11, model LSTM memberikan hasil yang lebih baik.

Perlu dipahami bahwa formula untuk menghitung metrik evaluasi, seperti MAE atau RMSE, melibatkan perhitungan selisih antara nilai asli dan prediksi. Jika nilai data asli dan prediksi memiliki skala yang besar, maka selisih antara keduanya juga akan menjadi besar. Misalnya, jika nilai asli adalah 10000 dan diprediksi 15000, maka selisihnya adalah 5000. Namun, jika nilai asli adalah 100 dan diprediksi 150, maka selisihnya hanya 50. Oleh karena itu, skala data menjadi salah satu faktor yang sangat mempengaruhi hasil dari metrik evaluasi.

Secara keseluruhan, hasil *output* model masih menunjukkan kualitas yang terbatas, terlihat dari evaluasi metrik yang telah diuji serta dari visualisasi hasil prediksi. Namun, penting untuk dicatat bahwa skala data, baik dari segi jumlah maupun nominalnya, turut berperan dalam memengaruhi performa metrik, seperti yang diamati pada intensitas cahaya rata-rata. Terlebih lagi jumlah dataset yang terbatas mempengaruhi proses pelatihan sehingga model tidak bisa mengenali kompleksitas dari data.

Dalam hal kualitas model, LSTM menunjukkan kualitas yang lebih unggul berkat kemampuan sel memori yang memungkinkannya untuk menangkap dan mengingat pola-pola yang kompleks. Meskipun hasil prediksi GRU cenderung lebih buruk, namun GRU menawarkan keunggulan dalam hal penyimpanan internal yang lebih sederhana, yang berarti membutuhkan sumber daya yang lebih sedikit [13]. Jika kebutuhan akan ukuran model yang ringan menjadi perhatian utama, GRU menjadi salah satu solusi yang lebih optimal.

### 3.2.10 Analisis Model Terbaik

Berdasarkan hasil dan pembahasan, maupun dari grafik ataupun tabel metrik model LSTM memberikan *output* yang lebih baik. LSTM lebih mampu melihat pola kompleks yang ada pada dataset. Untuk jumlah dataset sekitar 800 data, kemungkinan menggunakan model lainnya seperti regresi, selain itu bisa juga

dengan menambah jumlah data dalam proses pelatihan agar hasil prediksi model deep learning bisa lebih baik.

Tabel 3.12 Prevalensi hama

| Hama | Variabel | Kisaran | Rata-rata Jumlah |
|---|---|---|---|
| Lalat Buah | Temperatur (°C) | 20-32 | 165 |
| | Kelembapan (%) | 60-75 | 137 |
| | Curah Hujan (mm/Bulan) | 50-250 | 139 |
| | Intensitas Cahaya (Lux) | 500-2000 | 164 |
| Kutu Putih | Temperatur (°C) | 27-37 | 36 |
| | Kelembapan (%) | 50-60 | 36 |
| | Curah Hujan (mm/Bulan) | 0-225 | 28 |
| | Intensitas Cahaya (Lux) | 600-1000 | 46 |
| Kumbang Girang | Temperatur (°C) | 23-33 | 145 |
| | Kelembapan (%) | 75-85 | 145 |
| | Curah Hujan (mm/Bulan) | 0-225 | 28 |
| | Intensitas Cahaya (Lux) | 900-1700 | 63 |
| Tikus | Temperatur (°C) | 19-23 | 29 |
| | Kelembapan (%) | 40-70 | 19 |
| | Curah Hujan (mm/Bulan) | 25-200 | 17 |
| | Intensitas Cahaya (Lux) | 0 | 0 |
| Bajing | Temperatur (°C) | 19-23 | 29 |
| | Kelembapan (%) | 40-70 | 19 |
| | Curah Hujan (mm/Bulan) | 25-200 | 15 |
| | Intensitas Cahaya (Lux) | 0 | 0 |

Tabel 3.12 menjabarkan prevalensi hama dalam kisaran tertentu, model kecerdasan buatan diharapkan untuk memprediksi data cuaca dengan akurat agar ketika data yang diprediksi masuk kedalam kisaran dalam tabel maka bisa dipastikan kemunculan hama pada hari mendatang.

### 3.3 Kendala yang Ditemukan

Tantangan dalam magang ini mencakup kesulitan dalam menemukan model yang *robust* dan sesuai untuk digunakan pada dataset yang tersedia. Seringkali, model yang ditemukan tidak sepenuhnya cocok untuk karakteristik data yang ada.

Kurangnya jumlah data juga menjadi kendala utama, di mana pelatihan model hanya dapat dilakukan dengan 800 data yang didapatkan dari BMKG, dan 1200 data untuk intensitas cahaya. Jumlah data ini tergolong rendah untuk skala *deep learning*, dan dapat mengakibatkan kualitas akhir dari model yang kurang memuaskan.

Selain itu, keterbatasan pengetahuan juga menjadi hambatan dalam memahami dan mengembangkan model. Akses terbatas terhadap jurnal atau artikel berbayar juga membatasi sumber daya yang dapat digunakan sebagai referensi atau acuan dalam memecahkan masalah yang dihadapi.

### 3.4 Solusi atas Kendala yang Ditemukan

Dalam upaya untuk menemukan model yang cocok dengan data, eksperimentasi menjadi bagian penting dari proses pembuatan model. Eksperimen ini melibatkan berbagai konfigurasi dalam arsitektur model, termasuk penggunaan layer yang bervariasi serta penyesuaian parameter yang terkait. Setiap modifikasi yang dilakukan pada setiap layer perlu didokumentasikan dengan cermat, termasuk perubahan pada jumlah *layer*, jenis *layer* yang digunakan, dan jumlah *neuron* atau *unit*.

Permasalahan mengenai jumlah data akan bisa terselesaikan ketika sudah menggunakan data primer. Ketika sudah menggunakan modul *Internet of Things*, maka jumlah data akan bertambah tiap harinya, sehingga proses pelatihan dapat menggunakan data lebih banyak.

Mengatasi keterbatasan akses terhadap artikel berbayar dapat dilakukan dengan mencari artikel lain yang tersedia secara bebas atau yang memiliki akses terbuka. Meskipun konten dari artikel tersebut tidak selalu sesuai dengan yang diharapkan,

artikel-alternatif tersebut dapat memberikan wawasan tambahan atau perspektif baru yang berguna dalam memahami dan mengatasi permasalahan yang dihadapi.