

**PROYEKSI CURAH HUJAN DAERAH PADANG PARIAMAN**  
**MENGGUNAKAN DEEP LEARNING DENGAN**  
**METODE LONG SHORT-TERM MEMORY**

**SKRIPSI**

*Untuk Memenuhi Salah Satu Syarat  
Memperoleh Gelar Sarjana Komputer*

**Program Studi : Teknik INFORMATIKA**  
**Jenjang Pendidikan : Strata 1 (S1)**



**OLEH:**

**EDO SULAIMAN**  
**NIM. 18101152630092**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS PUTRA INDONESIA “YPTK” PADANG**

**2022**

## **BAB I**

### **PENDAHULUAN**

#### **1.1. Latar Belakang**

Dewasa ini data merupakan penunjang pengambilan keputusan secara cepat, atau dikenal dengan istilah *Data Driven Decision Making (DDDM)* di mana kemajuan teknologi berperan besar dalam memanfaatkan data dan informasi tersebut (Aditya, Mulyana, Eka, & Widianto, 2020). Dalam berbagai aspek termasuk melakukan prediksi mengenai informasi curah hujan yang akurat di mana pemodelan tersebut masih memiliki kekurangan seperti penggunaan jumlah *parameter*, asumsi matematis, dan rumusan persamaan yang cenderung rumit, untuk menghasilkan sebuah model prediksi yang mendekati keakuratan optimal harus memiliki banyak *parameter* dan *variabel input* untuk memenuhi sebuah asumsi prediksi (Supriyadi, 2019).

Mengatasi perihal tersebut, dikembangkanlah sebuah *Kecerdasan Buatan (Artificial Intelligence)* yang memiliki kemampuan untuk melakukan pembelajaran untuk menganalisis berbagai macam asumsi dan aspek yang berpengaruh untuk menarik kesimpulan (Supriyadi, 2019). *Artificial Intelligence (AI)* atau *kecerdasan buatan* didefinisikan secara berbeda dalam konteks yang berbeda pula dalam disiplin ilmu komputer AI adalah mempelajari cara menstimulasikan untuk melakukan tugas yang biasanya membutuhkan pemahaman seperti manusia (Knowledge@Wharton, 2018). AI juga suatu cabang ilmu komputer yang menggunakan lebih banyak simbol daripada angka, dan memproses informasi

berdasarkan jumlah aturan dalam merepresentasikan pengetahuan (Swarnkar & Swarnkar, 2019).

*Machine Learning* adalah bagian dari AI di mana mesin digunakan untuk belajar dari pengalaman masa lalu (Aditya, Mulyana, Eka, & Widianto, 2020). Algoritma *Machine Learning* digunakan dalam berbagai aplikasi, seperti dalam kedokteran, pengenalan email, pengenalan suara, dan visi komputer, di mana sulit atau tidak mungkin untuk mengembangkan algoritma konvensional untuk melakukan tugas yang diperlukan (Hu, Niu, Carrasco, Lennox, & Arvin, 2020).

Beberapa implementasi *Machine Learning* menggunakan data dan *Neural Network (Jaringan Saraf)* dengan cara yang meniru kerja otak biologis manusia (Zhou, 2019). *Deep Learning* dapat dipahami sebagai bentuk *Neural Network layer* berganda yang merupakan bagian dari *Machine Learning* yang dapat digunakan dalam tugas termasuk *computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection* dan *board game programs*, di mana mereka telah menghasilkan hasil yang sebanding dan dalam beberapa kasus melebihi kinerja para pakar (Hu, Niu, Carrasco, Lennox, & Arvin, 2020).

Misalnya, sebuah komputasi yang menggunakan *Deep Learning*, mampu memahami konsep seperti garis, bentuk, tekstur, dan juga pengaruhnya dengan melihat data-data citra tanpa bantuan tambahan dari manusia (Schneiderman & Kanade, 2002). *Machine Learning* senantiasa bekerja menggunakan 1 *layer* di mana *Deep Learning* bekerja lebih dari 1 *layer*. Untuk batasan *layer* dari *Deep Learning* itu sendiri sebagai *Neural Network* biasanya memiliki 3 *layer* atau lebih, makin

bayak *layer* yang digunakan akan memengaruhi lama waktu yang terpakai untuk komputer mengalkulasi (Hinton, et al., 2012).

*Layer* pada *Deep Learning* dapat di gambarkan seperti lapisan *neuron* pada otak manusia *layer* itu nantinya akan menggambarkan jarak atau vektor menggunakan Fungsi matematika yaitu fungsi *sigmoid* ( $\sigma$ ) (Putra J. G., 2020). Alasan fungsi *sigmoid* digunakan karena dalam fungsi ini membutuhkan perhitungan yang relatif mudah dan cepat. Selain itu, fungsi *sigmoid* dapat diartikan sebagai nilai peluang karena nilainya antara 0 dan 1 (Putra J. G., 2020).

Salah satu pendekatan *Deep Learning* yang mampu secara otomatis mempelajari fitur yang dideskripsikan dalam bentuk *vektor* adalah *Recurrent Neural Networks (RNN)* (Puspaningrum, Bunga, & Iryanto, 2020). Pada *RNN* sendiri teknik *learning* bekerja dengan menyimpan *layer* dari *output* kembali sebagai *input* pada *hidden layer* berikutnya hingga memprediksi hasil akhir (Tarkus, Sompie, & Jacobus, 2020). Kelemahan *RNN* adalah tidak mampu lagi untuk belajar menghubungkan informasi ketika ada kesenjangan yang terus tumbuh, memori yang tersimpan akan semakin tidak relevan seiring waktu berjalan karena tertimpa dengan memori baru (Putra, Osmond, & Ansori, 2020), di sebabkan kelemahan dari *RNN* sendiri tidak dapat mempelajari informasi yang terlalu jauh atau *Long-Term Dependencies*, yang cukup jauh pada masukannya (Wibisono & Khodra, 2018).

Penelitian *Deep Learning* terdahulu yang di lakukan Juanda, Jondri, & Rohmawati, tentang *Prediksi Harga Bitcoin Dengan Menggunakan Recurrent Neural Network* menurutnya, masalah prediksi *timeseries* adalah jenis pemodelan prediktif yang sulit, tidak seperti pemodelan prediktif regresi, *timeseries* juga

menambah kompleksitas ketergantungan urutan antar variabel *input*. *Recurrent Neural Network* terbukti berhasil digunakan untuk prediksi data *timeseries* karena *RNN* mampu menggunakan informasi yang telah direkam sebelumnya yang panjang urutannya atau *sequence*-nya beragam-ragam. Oleh karena itu, pembangunan sistem ini dibuat dengan metode *Recurrent Neural Network* dengan menggunakan algoritma *Backpropagation Through Time*. Hasil akhir Prediksi harga Bitcoin dapat dilakukan menggunakan Recurrent Neural Network. Akurasi rata-rata terbaik yang didapatkan adalah 98.76% pada data latih dan 97.46% pada data uji, dengan *parameter* jumlah pola *input* terbaik adalah 5, jumlah *epoch* 1000, nilai *learning rate* 0.001 dan jumlah *hidden unit* 50 (Juanda, Jondri, & Rohmawati, 2018).

*Long Short-Term Memory (LSTM)* merupakan sebuah pengembangan metode dari arsitektur *Recurrent Neural Network (RNN)*, Banyak peneliti yang mengembangkan metode *LSTM* di berbagai bidang seperti dalam bidang prediksi deret waktu atau *forecasting* dikarenakan metode *LSTM* mampu mengatasi kekurangan tersebut karena metode ini dapat mengatur memori pada setiap masukannya dengan menggunakan *memory cells* dan *gate units* pada setiap *neurons* yang berfungsi sebagai pengatur memori (Putra, Osmond, & Ansori, 2020). Contoh penggunaan *Deep Learning* untuk data *timeseries* yang banyak dihasilkan dari pengamatan cuaca adalah *LSTM*, *LSTM* sendiri diciptakan oleh Hochreiter dan Schmidhuber pada tahun 1997 (Supriyadi, 2019).

Penelitian *LSTM* terdahulu yang yang di teliti oleh Poornima & Pushpalatha, mengenai *Prediction of Rainfall Using Intensified LSTM Based Recurrent Neural Network with Weighted Linear Units* dalam penelitian tersebut menyajikan *Long Short-Term Memory (LSTM)* berbasis *Recurrent Neural Network (RNN)* untuk

memprediksi *curah hujan*. *Neural Network* dilatih dan diuji menggunakan kumpulan data standar curah hujan. Jaringan yang dilatih akan menghasilkan atribut prediksi curah hujan. *Parameter* yang dipertimbangkan untuk evaluasi kinerja dan efisiensi model prediksi curah hujan yang diusulkan adalah *Root Mean Square Error (RMSE)*, *akurasi*, *jumlah epoch*, *loss*, dan *learning rate*. Menurutnya *LSTM* dapat menyimpan data besar ke dalam memorinya dan dapat menghindari gradient yang hilang lebih baik dari pada *RNN* dan menunjukkan akurasi lebih baik dibandingkan *RNN*, *LSTM* pun juga mempertahankan akurasi di masa depan seiring dengan mempertimbangkan nilai *Root Mean Square Error (RMSE)*, *akurasi*, *jumlah epoch*, *loss*, dan *learning rate* (Poornima & Pushpalatha, 2019).

Penelitian *LSTM* terdahulu yang juga dilakukan oleh Supriyadi, mengenai metode *Deep Learning LSTM* untuk memprediksi *parameter* cuaca, seperti suhu udara, kelembaban, kecepatan angin, dan tekanan udara. Metode ini bekerja dengan memanfaatkan fungsi matematika seperti fungsi *tanh* dan *sigmoid* yang berada dalam *layer LSTM*. Adapun jumlah *layer* yang digunakan sebanyak 200 buah. Sedangkan jumlah datanya dibagi dua menjadi *training data* dan *test data* dengan rasio 9:1.pada bulan Januari 2019. Diperoleh *RMSE parameter* suhu udara, kelembaban, kecepatan angin, dan tekanan udara nilainya semakin baik ketika menggunakan *Deep Learning LSTM* dengan update dibandingkan *LSTM* tanpa update. Diperoleh hasil prediksi suhu udara, kelembaban, kecepatan angin, dan tekanan udara 1 hari ke depan memiliki *RMSE* yang baik. Dari *parameter* cuaca tersebut hanya *parameter* suhu dan kelembaban udara yang mengalami pertambahan *RMSE* seiring bertambahnya waktu. Sedangkan *parameter* kecepatan

angin dan tekanan udara mengalami penurunan di hari ketiga dan meningkat secara kontinu hingga 1 bulan ke depan (Supriyadi, 2019).

Berdasarkan rincian penjelasan sebelumnya, sangat dimungkinkan untuk menggunakan *Deep Learning* dengan metode *LSTM* dikarenakan mendukung kegiatan proyeksi curah hujan. Karena data pengamatan meteorologi umumnya berupa *vektor* dan *timeseries*, untuk itu peneliti membuat Penelitian dalam bentuk skripsi dengan judul “**PROYEKSI CURAH HUJAN DAERAH PADANG PARIAMAN MENGGUNAKAN DEEP LEARNING DENGAN METODE LONG SHORT-TERM MEMORY**”.

## **1.2. Perumusan Masalah**

Berdasarkan uraian pada latar belakang masalah maka yang menjadi perumusan masalah pada skripsi ini adalah sebagai berikut :

1. Bagaimana melakukan proses prediksi curah hujan dengan menggunakan pendekatan *Deep Learning* menggunakan metode *Long Short-Term Memory* dapat menghasilkan alternatif dalam pengambilan sebuah keputusan ?
2. Bagaimana penerapan *Deep Learning* menggunakan metode *Long Short-Term Memory* dapat melakukan prediksi curah hujan.
3. Bagaimana pengujian *Deep Learning* dengan metode *Long Short-Term Memory* di implementasikan ke dalam sebuah sistem yang dibangun untuk memprediksi curah hujan di daerah padang Pariaman ?

### 1.3. Hipotesis

Berdasarkan rumusan masalah yang telah ditentukan maka dapat diambil kesimpulan sebagai berikut :

1. Dengan melakukan proses prediksi curah hujan dengan menggunakan pendekatan *Deep Learning* diharapkan dapat menghasilkan alternatif dalam pengambilan sebuah keputusan.
2. Dengan menggunakan konsep pendekatan *Deep Learning* metode *Long Short-Term Memory* diharapkan dapat melakukan prediksi curah hujan.
3. Penerapan *Deep Learning* dengan metode *Long Short-Term Memory* diharapkan dapat di implementasikan ke dalam sebuah sistem yang dibangun untuk memprediksi curah hujan di daerah padang Pariaman.

### 1.4. Batasan Masalah

Adapun batasan masalah pada penelitian ini adalah sebagai berikut :

1. Data yang digunakan adalah data *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* dari tahun 1985 sampai tahun 2021.
2. Metode yang digunakan dalam penelitian ini *Long Short-Term Memory*.
3. Menggunakan Bahasa Pemograman *Python*.
4. Fungsi Aktivasi yang digunakan adalah fungsi *Sigmoid* ( $\sigma$ ) dan *Tanh*.
5. Untuk fitur variabel yang digunakan dalam penelitian ini hanya terdiri dari curah hujan (rr).
6. Data dibagi menjadi dua bagian, yaitu data *training* dan data *testing* dengan rasio 9:1 di mana 9 untuk *training* dan 1 untuk *testing*.

## 1.5. Tujuan Penelitian

Adapun Tujuan dari penelitian ini adalah sebagai berikut :

1. Menerapkan *Deep Learning* menggunakan metode *Long Short-Term Memory* untuk melakukan proses prediksi curah hujan dalam menghasilkan alternatif dalam pengambilan sebuah keputusan.
2. Menerapkan pendekatan *Deep Learning* menggunakan metode *Long Short-Term Memory* untuk melakukan prediksi curah hujan.
3. Melakukan pengujian *Deep Learning* dengan metode *Long Short-Term Memory* di implementasikan ke dalam sebuah sistem yang dibangun untuk memprediksi curah hujan di daerah padang Pariaman.

## 1.6. Manfaat Penelitian

Hasil dari penelitian di harapkan dapat memberikan manfaat sebagai berikut :

1. Menambah pengetahuan peneliti dalam memprediksi / memproyeksi data berbentuk *timeseries* yang di selesaikan dengan cara *Deep Learning* dengan menggunakan metode *Long Short-Term Memory*.
2. Membuktikan keakuratan metode *Long Short-Term Memory* dalam melakukan peramalan khususnya curah hujan.
3. Memberikan informasi tambahan mengenai peramalan curah hujan di daerah padang Pariaman yang akan terjadi pada masa mendatang.

## 1.7. Gambaran Umum Objek Penelitian

*Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* merupakan BMKG Stasiun Klimatologi Padang Pariaman yang terletak di daerah *Jalan Raya Padang* –

*Bukittinggi KM. 51 Kapalo Hilalang Sumatera Barat.* Berikut ini adalah gambaran umum tentang Objek Penelitian :

### **1.7.1. Sejarah BMKG**

Pengamatan meteorologi dan geofisika di Indonesia dimulai pertama kali pada tahun 1841 diawali dengan pengamatan yang dilakukan secara individual oleh Dr. Pieter Loth Onnen, Kepala Rumah Sakit di Bogor. Dari tahun ke tahun kegiatannya berkembang seiring dengan meningkatnya kebutuhan akan data hasil pengamatan cuaca dan geofisika. Pada tahun 1866, pemerintah Hindia Belanda meresmikan kegiatan pengamatan perorangan sebagai lembaga pemerintah dengan nama Observatorium Magnetisch en Meteorologisch atau Observatorium Magnetik dan Meteorologi yang dipimpin oleh Dr. Pieter Adrian Bergsma.

Pada tahun 1879, 74 jaringan alat pengukur hujan dibangun di Jawa. pengamatan medan magnet bumi dipindahkan dari Jakarta ke Bogor Pada tahun 1902. Pada tahun 1908 pemantauan gempa dimulai dengan pemasangan komponen horizontal seismograf Wiechert di Jakarta, sedangkan pemasangan komponen vertikal dilakukan pada tahun 1928. Pada tahun 1912, pengamatan meteorologi ditata ulang dengan menambahkan jaringan sekunder. Sedangkan pada tahun 1930 jasa meteorologi mulai digunakan untuk penerangan.

Pada masa pendudukan Jepang antara tahun 1942 dan 1945, nama badan meteorologi dan geofisika diubah menjadi Kisho Kauso Kusho atau Lembaga Meteorologi. Pada tahun 1945 Setelah proklamasi kemerdekaan Indonesia, badan tersebut dibagi menjadi dua: Di Yogyakarta dibentuk Badan Meteorologi yang berkedudukan di lingkungan Mabes TNI khusus untuk melayani kepentingan

Angkatan Udara. Di bawah Kementerian Pekerjaan Umum dan Energi Badan Meteorologi dan Geofisika dibentuk Di Jakarta.

Pada tanggal 21 Juli 1947 Biro Meteorologi dan Geofisika diambil alih oleh Pemerintah Belanda dan namanya diubah menjadi Meteorologisch en Geofisiche Dienst. Badan Meteorologi dan Geofisika yang dikelola oleh Pemerintah Republik Indonesia berada di Jl. Gondangdia, Jakarta. Pada tahun 1949, setelah penyerahan kedaulatan Republik Indonesia dari Belanda, Meteorologisch en Geofisiche Dienst diubah menjadi Biro Meteorologi dan Geofisika di bawah Departemen Perhubungan dan Pekerjaan Umum. Selanjutnya pada tahun 1950 Indonesia resmi masuk sebagai anggota Organisasi Meteorologi Dunia dan Kepala Badan Meteorologi dan Geofisika menjadi Wakil Tetap Indonesia dengan WMO.

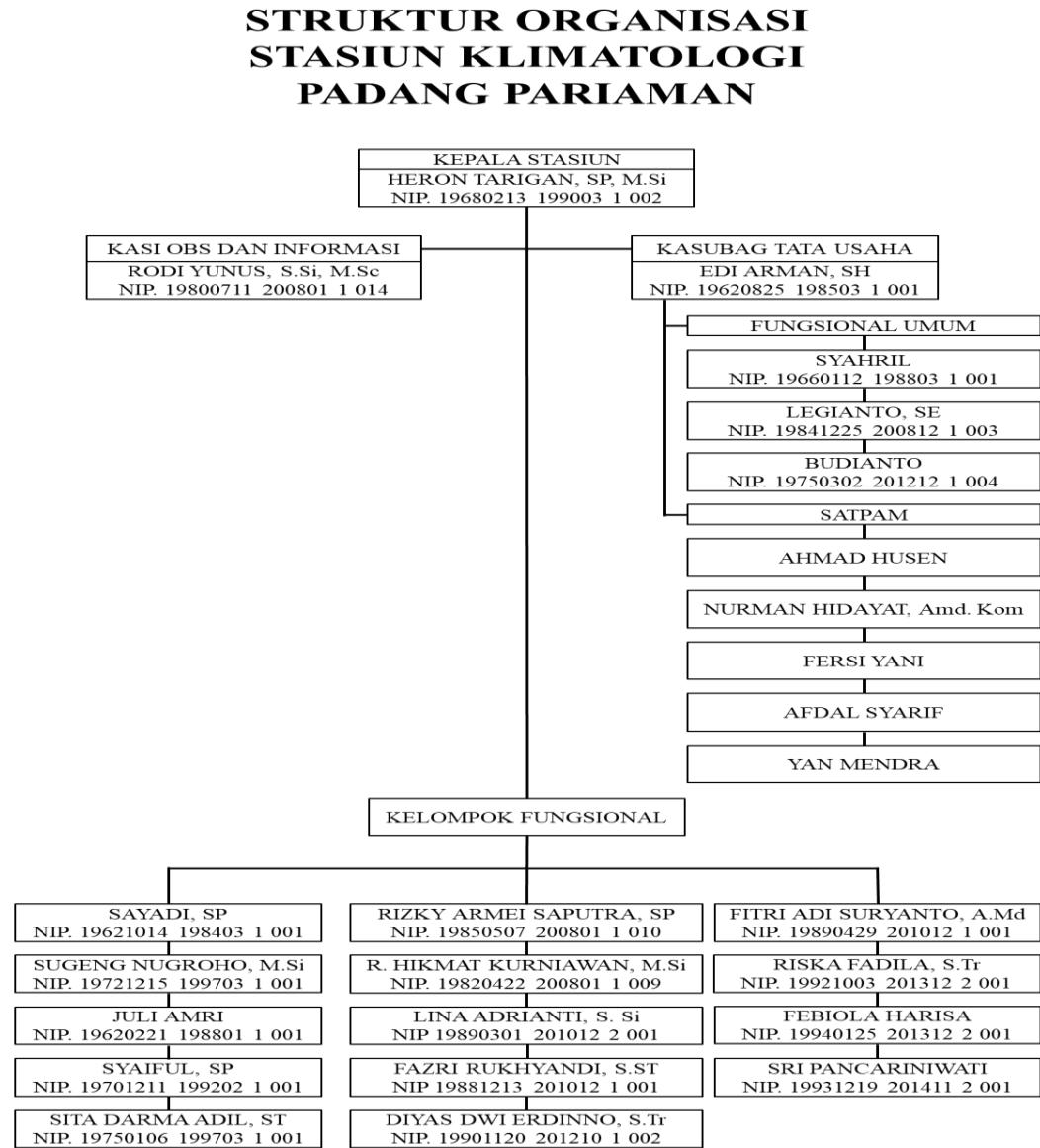
Pada tahun 1955 Biro Meteorologi dan Geofisika beubah nama menjadi Lembaga Meteorologi dan Geofisika di bawah Kementerian Perhubungan, dan pada tahun 1960 namanya dikembalikan menjadi Biro Meteorologi dan Geofisika di bawah Kementerian Perhubungan Udara.

Pada tahun 1965, namanya diubah menjadi Direktorat Meteorologi dan Geofisika, posisinya tetap di bawah Kementerian Perhubungan Udara. statusnya dinaikkan menjadi lembaga setingkat eselon I dengan nama Badan Meteorologi dan Geofisika, dengan jabatan tetap di bawah Kementerian Perhubungan. Badan Meteorologi dan Geofisika.

Melalui Peraturan Presiden Nomor 61 Tahun 2008, BMG berganti nama menjadi Badan Meteorologi, Klimatologi, dan Geofisika dengan status tetap sebagai Lembaga Pemerintah Non Departemen.

### 1.7.2. Struktur Organisasi

Berikut adalah bentuk struktur organisasi BMKG Padang Pariaman :



(Sumber : Sta. Klim. Kelas II Padang Pariaman, 2021)

**Gambar 1.1. Struktur Organisasi BMKG Padang Pariaman**

*Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* dipimpin oleh Kepala Stasiun Klimatologi (Kaslim) bertanggung jawab terhadap seluruh bidang

yang ada pada instansi BMKG, Staf yang membantu Kaslim dalam menjalankan aktivitas di kantor adalah sebagai berikut :

1. Tata usaha yang bertanggung jawab terhadap *administrasi kantor*.
2. Bagian analisa yang bertanggung jawab terhadap pengolahan data dan analisis data-data yang dikirim ke Balai Wilayah I.
3. Tenaga teknis yang bertanggung jawab terhadap data-data klimatologi yang ada di *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman*.
4. Tim pengamat yang bertanggung jawab terhadap pengaturan jadwal pengamatan di *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman*.
5. Tim komunikasi dan peralatan yang bertanggung jawab terhadap pengiriman informasi kondisi peralatan yang ada di *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman*.

### **1.7.3. Visi**

Mewujudkan BMKG yang handal, tanggap dan mampu dalam rangka mendukung keselamatan masyarakat serta keberhasilan pembangunan nasional, dan berperan aktif di tingkat Internasional.

Terminologi di dalam visi tersebut dapat dijelaskan sebagai berikut :

1. Pelayanan informasi meteorologi, klimatologi, kualitas udara, dan geofisika yang handal ialah pelayanan BMKG terhadap penyajian data, informasi pelayanan jasa meteorologi, klimatologi, kualitas udara, dan geofisika yang akurat, tepat sasaran, tepat guna, cepat, lengkap, dan dapat dipertanggungjawabkan.

2. Tanggap dan mampu dimaksudkan BMKG dapat menangkap dan merumuskan kebutuhan stakeholder akan data, informasi, dan jasa meteorologi, klimatologi, kualitas udara, dan geofisika serta mampu memberikan pelayanan sesuai dengan kebutuhan pengguna jasa.

#### **1.7.4. Misi**

Dalam rangka mewujudkan Visi BMKG, maka diperlukan visi yang jelas yaitu berupa langkah-langkah BMKG untuk mewujudkan Misi yang telah ditetapkan yaitu :

1. Mengamati dan memahami fenomena meteorologi, klimatologi, kualitas udara dan geofisika.
2. Menyediakan data, informasi dan jasa meteorologi, klimatologi, kualitas udara dan geofisika yang handal dan terpercaya.
3. Mengkoordinasikan dan memfasilitasi kegiatan di bidang meteorologi, klimatologi , kualitas udara dan geofisika.
4. Berpartisipasi aktif dalam kegiatan internasional di Bidang meteorologi, klimatologi, kualitas udara dan geofisika.

## BAB II

### LANDASAN TEORI

#### 2.1. Rekayasa Perangkat Lunak

IEEE Computer Society mendefinisikan Rekayasa Perangkat Lunak (RPL) sebagai penerapan suatu pendekatan yang sistematis, disiplin dan terkuantifikasi atas pengembangan, penggunaan dan pemeliharaan perangkat lunak, serta studi atas pendekatan-pendekatan ini, yaitu penerapan pendekatan engineering atas perangkat lunak (Hasanah & Untari, 2020).

RPL sendiri adalah suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, desain, penggodaan, pengujian sampai pemeliharaan sistem setelah digunakan (Hasanah & Untari, 2020).

RPL lebih fokus pada praktik pengembangan perangkat lunak dan mengirimkan perangkat lunak yang bermanfaat kepada pelanggan (customer). Adapun ilmu komputer lebih fokus pada teori dan konsep dasar perangkat komputer. Rekayasa perangkat lunak lebih fokus pada bagaimana membuat perangkat lunak yang memenuhi kriteria berikut (Hasanah & Untari, 2020) :

- a) Dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring berkembangnya teknologi dan lingkungan (maintainability)
- b) Dapat diandalkan dengan proses bisnis yang dijalankan peubahannya yang terjadi (dependability robust)
- c) Efisien dari segi sumber daya dan penggunaan
- d) Kemampuan untuk dipakai sesuai dengan kebutuhan (usability)

### **2.1.1. Proses Rekayasa Perangkat Lunak**

Kerangka kerja proses membangun dasar bagi proses rekayasa perangkat lunak yang lengkap dengan cara mengidentifikasi sejumlah kecil aktivitas kerangka kerja yang cocok bagi semua proyek rekayasa perangkat lunak (Setiyani, 2018). Kerangka kerja proses pada rekayasa perangkat lunak terdiri atas lima aktivitas berikut (Setiyani, 2018) :

#### **2.1.1.1. Komunikasi**

Komunikasi, bertujuan untuk memahami tujuan-tujuan stakeholder atas proyek perangkat lunak yang sedang dikembangkan dan mengumpulkan kebutuhan-kebutuhan yang akan membantu mendefinisikan fitur-fitur perangkat lunak berikut dengan fungsi-fungsinya (Setiyani, 2018) :

#### **2.1.1.2. Perencanaan**

Kegiatan perencanaan menciptakan suatu peta yang dapat membantu membimbing tim perangkat lunak. Rencana proyek perangkat lunak menggambarkan risiko – risiko yang mungkin muncul, sumber daya yang akan dibutuhkan, produk – produk kerja yang harus dihasilkan dan schedule kerja (Setiyani, 2018) :

#### **2.1.1.3. Pemodelan**

Pemodelan, dilakukan bertujuan untuk membuat sketsa sehingga tim perangkat lunak dapat memahami gambaran besar produk yang akan di buat (Setiyani, 2018) :

#### **2.1.1.4. Konstruksi**

Konstruksi, sendiri adalah kegiatan yang menggabungkan penggodaan dan pengujian (Setiyani, 2018) :

### **2.1.1.5. Penyerahan**

Penyerahan di sini merupakan Penyerahan perangkat lunak kepada *user*, penyajian perangkat lunak kepada *user* untuk di evaluasi (Setiyani, 2018) :

### **2.1.2. *Software Development Life Cycle (SDLC)***

System Development Life Cycle (SDLC) adalah metodologi klasik yang digunakan untuk mengembangkan, memelihara dan menggunakan sistem informasi. Siklus hidup sistem itu sendiri merupakan metodologi, tetapi polanya lebih dipengaruhi oleh kebutuhan untuk mengembangkan sistem yang lebih cepat. Pengembangan sistem yang lebih cepat dapat dicapai dengan peningkatan siklus hidup dan penggunaan peralatan pengembangan berbasis komputer (Wahyudi, 2018).

#### **2.1.2.1. Tahap-Tahap SDLC**

Secara umum tahap-tahap dalam System Development Life Cycle (SDLC) terbagi dalam beberapa tahap (Wahyudi, 2018):

##### **2.1.2.1.1. Tahap Perencanaan Sistem (*system planning*)**

Tahap Planning Merupakan tahap awal dari pengembangan sistem, tahap ini bertujuan untuk mengidentifikasi dan memprioritaskan sistem informasi apa yang akan dikembangkan, sasaran-sasaran yang ingin dicapai, jangka waktu pelaksanaan serta mempertimbangkan dana yang tersedia dan siapa yang melaksanakan (Wahyudi, 2018).

##### **2.1.2.1.2. Tahap Analisis Sistem (*system analysis*)**

Tahap Analisis Sistem adalah penelitian atas sistem yang telah ada dengan tujuan untuk merancang sistem baru atau memperbaharui sistem yang sudah ada (Wahyudi, 2018).

#### ***2.1.2.1.3. Tahap Perancangan/Desain Sistem (system design)***

Tahap Rancangan sistem adalah penentuan proses dan data yang diperlukan oleh sistem baru. Jika sistem ini berbasis komputer, rancangan dapat menyertakan spesifikasi jenis peralatan yang akan digunakan (Wahyudi, 2018).

#### ***2.1.2.1.4. Tahap Penerapan/Implementasi Sistem (system implementation)***

Tahap Penerapan merupakan kegiatan memperoleh dan mengintegrasikan sumber daya fisik dan konseptual yang menghasilkan suatu sistem yang bekerja. Pada tahapan ini dilakukan beberapa hal yaitu: Coding, Testing, Instalasi. Dan *Output* dari tahapan ini adalah : source code, prosedur, pelatihan (Wahyudi, 2018).

#### ***2.1.2.1.5. Tahap Pemeliharaan/Perawatan Sistem***

Tahap pemeliharaan/perawatan sistem merupakan tahap yang dilakukan setelah tahap implementasi yang meliputi penggunaan sistem, audit sistem, penjagaan sistem, perbaikan sistem dan peningkatan sistem (Wahyudi, 2018).

### **2.1.2.2. Model Waterfall**

Model SDLC air terjun (waterfall) sering juga disebut model sequential linier (sequential linier) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sequential atau terurut dimulai dari analisis kebutuhan perangkat lunak, desain, pembuatan kode program, pengujian, dan pemeliharaan (maintenance) (Tabrani & Pudjiarti, 2021). Berikut tahap tahapannya sebagai berikut (Tabrani & Pudjiarti, 2021):

#### ***2.1.2.2.1. Analisis Kebutuhan Perangkat Lunak***

Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang

dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan (Tabrani & Pudjiarti, 2021).

#### **2.1.2.2.2. Desain**

Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur penggodaan. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan (Tabrani & Pudjiarti, 2021).

#### **2.1.2.2.3. Pembuatan Kode Program**

Desain harus ditransaksikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain (Tabrani & Pudjiarti, 2021).

#### **2.1.2.2.4. Pengujian**

Pengujian fokus pada perangkat lunak secara dari segi logic dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalkan kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan (Tabrani & Pudjiarti, 2021).

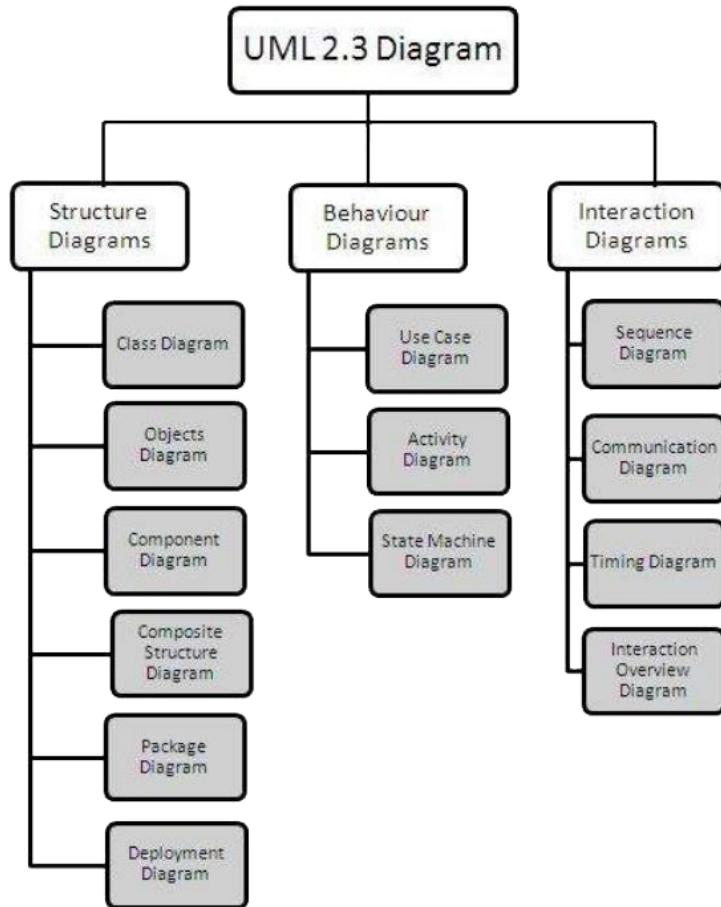
#### **2.1.2.2.5. Pendukung atau Pemeliharaan (*maintenance*)**

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi

dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru (Tabrani & Pudjiarti, 2021).

## **2.2. Unified Modelling Language (*UML*)**

Pemodelan dalam suatu rekayasa perangkat lunak merupakan suatu hal yang dilakukan di tahapan awal. Pemodelan dalam perangkat lunak merupakan suatu yang harus dikerjakan di bagian awal dari rekayasa, dan pemodelan ini akan mempengaruhi perkerjaan-pekerjaan dalam rekayasa perangkat lunak tersebut. Salah satu perangkat pemodelan adalah Unified Modelling Language (UML). UML merupakan salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan mendokumentasikan sistem perangkat lunak (Hasanah & Untari, 2020).



(Sumber : Hasanah & Untari, 2020).

### **Gambar 2.1. Bagan UML**

Berdasarkan Gambar 2.1 berikut penjelasan singkat dari pembagian kategori tersebut (Hasanah & Untari, 2020) :

- a) Behavior diagram, merupakan kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada suatu sistem.
- b) Interaction diagram, merupakan kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar sub sistem pada suatu sistem.

- c) Structure diagram, merupakan kumpulan diagram yang digunakan untuk menggambarkan struktur statis dari sistem yang dimodelkan.

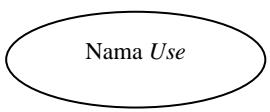
### **2.2.1. Behavior Diagrams**

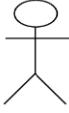
Behavior diagram adalah salah satu jenis diagram yang ada di dalam *Unified Modeling Language* (UML), di mana diagram ini digunakan untuk memberikan gambaran tingkah laku sebuah sistem informasi dan bagaimana sistem informasi tersebut melakukan tindakan terhadap kejadian atau perubahan (Sulistyo, Yudhana, & Sunardi, 2018).

#### **2.2.1.1. Use case Diagram**

*Use case* diagram adalah teknik untuk merekam persyaratan fungsional sebuah sistem, menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. *Use case* diagram menekankan kepada “apa” yang diperlukan oleh sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Seorang atau sebuah aktor adalah sebuah entitas dapat berupa manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. (Hasanah & Untari, 2020).

**Tabel 2.1. Notasi Use case Diagram**

No	Simbol	Keterangan
1	<i>Use case</i> 	Fungsionalitas yang disediakan sistem sebagai <i>unit-unit</i> yang saling bertukar pesan antar <i>unit</i> atau aktor.

2	<i>Actor</i> 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem itu.
3	<i>Association</i> 	<i>use case</i> yang memiliki interaksi dengan aktor.
4	<i>Extend</i> 	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> di mana <i>use case</i> yang ditambahkan dapat berdiri sendiri.
5	<i>Generalization</i> 	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> di mana <i>use case</i> yang ditambahkan memerlukan ini untuk menjalankan fungsinya.

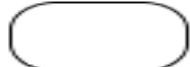
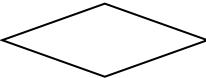
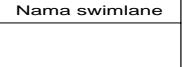
(Sumber : Hasanah & Untari, 2020)

### 2.2.1.2. Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram digunakan untuk menggambarkan langkah-langkah atau aktivitas pada suatu sistem (Hasanah & Untari, 2020).

**Tabel 2.2. Notasi Activity Diagram**

No	Simbol	Keterangan
1	<i>Initial State</i> 	Sebuah diagram aktivitas memiliki sebuah status awal

2	<i>Activity</i> 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
3	<i>Decision</i> 	Asosiasi percabangan di mana jika ada pilihan aktivitas lebih dari satu
4	<i>Join</i> 	Asosiasi penggabungan di mana lebih dari satu aktivitas digabungkan menjadi satu
5	<i>Final State</i> 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6	<i>Swimlane</i>  or 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

(Sumber : Hasanah & Untari, 2020)

### 2.2.1.3. *Statechart Diagram*

*Statechart* diagram menelusuri individu-individu objek melalui keseluruhan daur hidupnya, menypesifikasi semua urutan yang mungkin dari pesan-pesan yang akan diterima objek tersebut, bersama-sama dengan tanggapan atas pesan-pesan tersebut (Ilham & Fajri, 2020).

**Tabel 2.3. Notasi Statechart Diagram**

No	Simbol	Keterangan
1	<i>Initial State</i> 	Kelas pada struktur sistem
2	<i>Final State</i> 	Sama dengan konsep <i>interface</i> dalam beberapa pemrograman berrorientasi objek.
3	<i>Event</i> 	Relasi antar kelas dengan makna umum. Asosiasi biasanya juga disertai dengan multiplicity
4	<i>State</i> 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.

(Sumber : Ilham & Fajri, 2020).

## 2.2.2. Interaction Diagrams

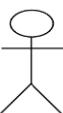
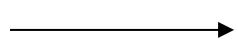
Interaction diagram merupakan diagram yang digunakan untuk menggambarkan bagaimana sebuah objek berinteraksi baik aktor dan objek sistem (Epandi, Panjaitan, & Yulianingsih, 2018).

### 2.2.2.1. Sequence Diagram

*Sequence* diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence* diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan *output* tertentu.

Diawali dari apa yang memicu aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan (Rinaldi, 2019).

**Tabel 2.4. Notasi *State Machine Diagram***

No	Simbol	Keterangan
1	<p><i>Actor</i></p>  <p>Or</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <u>Nama aktor</u> </div>	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi biasanya dinyatakan menggunakan kata benda di awal frasa nama actor
2	<p><i>Lifeline</i></p> 	Menyatakan kehidupan suatu objek
3	<p><i>Object</i></p> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <u>objek : kelas</u> </div>	Menyatakan objek yang berinteraksi pesan
4	<p><i>Active Time</i></p> 	Menyatakan objek dalam keadaan aktif berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya,
5	<p><i>Message type Create</i></p> <p><i>&lt;&lt;create&gt;&gt;</i></p> 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

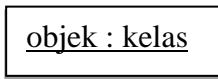
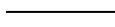
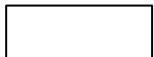
6	<i>Message type Call</i> 1:nama_metode() →	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri
7	<i>Message type Send</i> 1: masukan →	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim
8	<i>Message type Return</i> 1: keluaran →	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
9	<i>Message type Destroy</i> <<destroy>> → 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, jika ada create maka ada destroy.

(Sumber : Rinaldi, 2019)

### 2.2.2.2. Collaboration Diagram

*Collaboration* diagram adalah cara alternatif untuk mengetahui tahap-tahap terjadinya suatu aktivitas. Perbedaan antara *Collaboration* dan *sequence* diagram adalah *Collaboration* diagram memperlihatkan bagaimana hubungan antara beberapa objek berdasarkan urutan dari pesan, sedangkan *sequence* diagram memperlihatkan bagaimana urutan kejadian berdasarkan waktu (Ilham & Fajri, 2020).

**Tabel 2.5. Notasi Collaboration Diagram**

No	Simbol	Keterangan
1	<i>Object</i> 	Objek yang melakukan interaksi pesan.
2	<i>Link</i> 	Relasi antara objek yang menghubungkan objek satu dengan lainnya atau dengan diri sendiri.
4	<i>State</i> 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.

(Sumber : Ilham & Fajri, 2020)

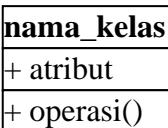
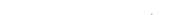
### 2.2.3. Structure Diagrams

Structure diagram merupakan diagram yang digunakan untuk menggambarkan struktur dari perangkat lunak atau sistem yang dikembangkan (Ependi, Panjaitan, & Yulianingsih, 2018).

#### 2.2.3.1. Class Diagram

*Class* diagram adalah sebuah spesifikasi yang jika di instansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berrorientasi objek. *Class* diagram menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi). *Class* diagram menggambarkan struktur dan deskripsi *class*, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain (Rinaldi, 2019).

**Tabel 2.6. Notasi Class Diagram**

No	Simbol	Keterangan
1	<i>Class</i> 	Kelas pada struktur sistem
2	<i>Interface</i> 	Sama dengan konsep <i>interface</i> dalam beberapa pemrograman berorientasi objek.
3	<i>Association</i> 	Relasi antar kelas dengan makna umum. Asosiasi biasanya juga disertai dengan multiplicity
4	<i>Directed Association</i> 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
5	<i>Dependency</i> 	Relasi antar kelas dengan ketergantungan antar kelas
6	<i>Aggregation</i> 	Relasi antar kelas dengan makna semua-bagian (whole-part)

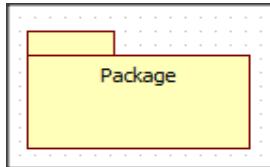
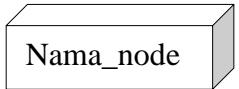
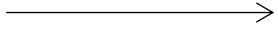
(Sumber : Rinaldi, 2019)

### 2.2.3.2. Deployment Diagram

Menggambarkan secara lengkap bagaimana komponen di *deployment* dalam infrastruktur sistem, di mana komponen akan terletak, bagaimana kemampuan jaringan pada kondisi tertentu, spesifikasi *server*, dan hal-hal lain yang bersifat fiskal (Wijaya, Masriadi, & Ikhlas, 2020).

**Tabel 2.7. Notasi Deployment Diagram**

No	Simbol	Keterangan
----	--------	------------

1	<i>Package</i> 	Package merupakan sebuah bungkusan dari satu atau lebih komponen.
2	<i>Node</i> 	Biasanya mengacu pada perangkat keras( <i>hardware</i> ), perangkat lunak yang tidak dibuat sendiri ( <i>software</i> ), jika di dalam node disertakan komponen untuk mengonsistensikan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen.
4	<i>Dependency</i> 	Ketergantungan antar komponen, arah panah mengarah pada komponen yang dipakai.
5	<i>Link</i> 	Relasi antar komponen.

(Sumber : Wijaya, Masriadi, & Ikhlas, 2020).

### 2.3. Artificial Intelligence

*Andreas Kaplan* dan *Michael Haenlein* mendefinisikan kecerdasan buatan / *Artificial Intelligence (AI)* sebagai kemampuan sistem untuk menafsirkan data eksternal dengan benar, untuk belajar dari data tersebut, dan menggunakan pembelajaran tersebut guna mencapai tujuan dan tugas tertentu melalui adaptasi yang fleksibel. Sistem seperti ini umumnya dianggap sebagai komputer (Siahaan, et al., 2020).

Lebih lanjut *Budiharto* menyatakan bahwa *Intelligence* merupakan istilah yang kompleks yang dapat didefinisikan dengan ungkapan yang berbeda seperti logika, pemahaman, self-awareness, pembelajaran, perencanaan, dan problem solving. Sedangkan “Artificial” adalah sesuatu yang tidak nyata, seperti tipuan karena merupakan hasil simulasi (*Sihombing & Syaputra*, 2020).

*Savitri* menguraikan bahwa Kecerdasan buatan / Artificial intelligence (AI) merupakan bidang ilmu komputer yang menekankan pada penciptaan mesin cerdas yang bekerja dan bereaksi seperti manusia yang perkembangannya terjadi sangat pesat di era revolusi industri keempat (*Sihombing & Syaputra*, 2020).

Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin / komputer agar dapat melakukan pekerjaan seperti yang dapat dilakukan oleh manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, permainan komputer (games), logika fuzzy, jaringan saraf tiruan dan robotika (*Siahaan, et al.*, 2020).

Tujuan utama dari pembuatan AI adalah untuk membuat mesin memiliki fungsi yang memiliki kriteria-kriteria kecerdasan di dalamnya sehingga mesin tersebut mampu melakukan pekerjaan manusia yang lebih kompleks, tergantung dari tingkat kecerdasan AI yang digunakan (*Gunova*, 2021).

Dalam penerapannya, terdapat 6 kemampuan utama yang dapat diklasifikasikan sebagai AI (*Gunova*, 2021), Kemampuan tersebut antara lain :

- Representasi Informasi/Pengetahuan (Knowledge Representation)
- Perencanaan (Planning)
- Persepsi (Perception)

- *Machine Learning / Deep Learning*
- Pemahaman Bahasa (Natural Language Processing)
- Robotics

Meskipun 6 kemampuan di atas memiliki fungsi yang berbeda-beda, kemampuan tersebut saling berhubungan satu sama lain saat pengaplikasiannya. Bahkan hubungan antar kemampuan utama di atas akan membentuk kemampuan baru (Gunova, 2021).

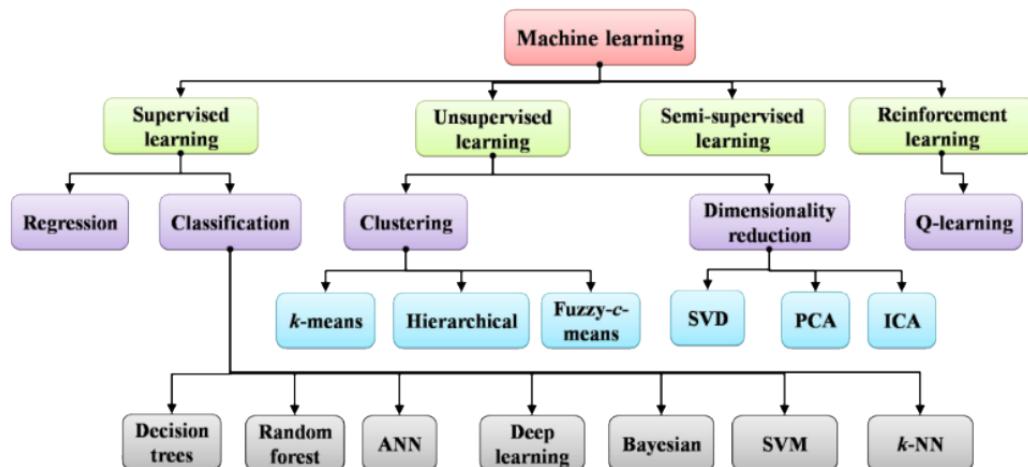
#### **2.4. *Machine Learning***

*Machine Learning* merupakan Disiplin ilmu yang menggunakan berbagai pendekatan untuk mengajarkan komputer untuk menyelesaikan tugas-tugas di mana tidak ada algoritma yang sepenuhnya memuaskan tersedia. Dalam kasus di mana terdapat sejumlah besar jawaban potensial, satu pendekatan adalah memberi *label* beberapa jawaban yang benar sebagai valid. Ini kemudian dapat digunakan sebagai data pelatihan bagi komputer untuk meningkatkan algoritme yang digunakannya untuk menentukan jawaban yang benar. Misalnya, untuk melatih sistem untuk tugas pengenalan karakter digital, kumpulan data MNIST dari angka tulisan tangan sering digunakan (Alpaydin, 2020).

Program *Machine Learning* dapat melakukan tugas tanpa diprogram secara eksplisit untuk melakukannya. Ini melibatkan komputer belajar dari data yang disediakan sehingga mereka melakukan tugas-tugas tertentu. Untuk tugas-tugas sederhana yang ditugaskan ke komputer, dimungkinkan untuk memprogram algoritme yang memberi tahu mesin bagaimana menjalankan semua langkah yang diperlukan untuk memecahkan masalah yang dihadapi; di bagian komputer, tidak

diperlukan pembelajaran. Untuk tugas yang lebih maju, mungkin sulit bagi manusia untuk membuat algoritme yang diperlukan secara manual. Dalam praktiknya, ternyata lebih efektif untuk membantu mesin mengembangkan algoritmenya sendiri, daripada meminta pemrogram manusia menentukan setiap langkah yang diperlukan (Alpaydin, 2020).

Algoritma *Machine Learning* digunakan dalam berbagai macam aplikasi, seperti dalam kedokteran, penyaringan email, pengenalan suara, dan visi komputer, di mana sulit atau tidak mungkin untuk mengembangkan algoritma konvensional untuk melakukan tugas-tugas yang diperlukan (Hu, Niu, Carrasco, Lennox, & Arvin, 2020).



(Sumber : Kumar, Amgoth, & Annavarapu, 2019)

**Gambar 2.2. Struktur *Machine learning***

#### 2.4.1. Tipe-tipe *Machine learning*

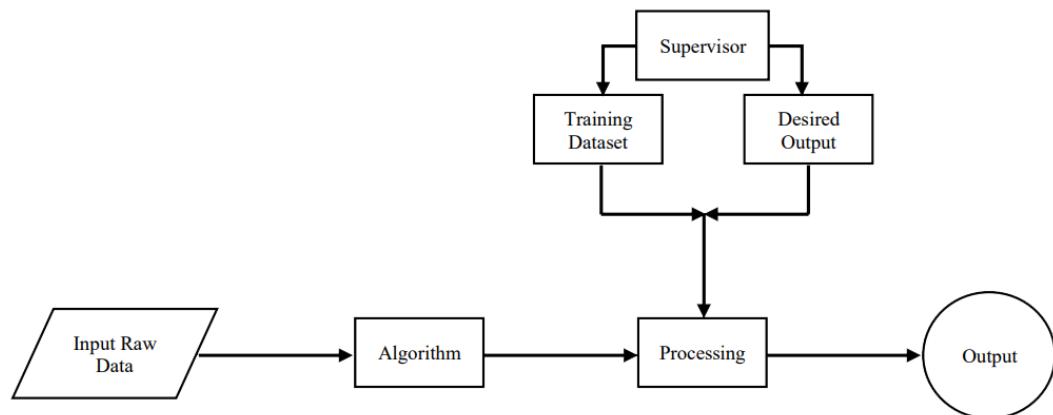
Dalam *Machine Learning*, terdapat beberapa metode atau pendekatan yang dapat digunakan mesin untuk melakukan pembelajaran. Namun secara garis besar,

terdapat 3 metode *Machine Learning* yang dipisahkan berdasarkan tipe *input* atau dataset dan cara pelatihannya (Gunova, 2021). Metode-metode tersebut antara lain :

#### **2.4.1.1. Supervised learning**

Supervised *Learning* (SL) merupakan sebuah metode ML di mana metode ini memberikan kumpulan data yang berlabel dan data *input* ke dalam mesin. Maksud dari dataset yang berlabel ini ialah untuk setiap tipe/bentuk kumpulan data yang telah diberikan, *outputnya* telah ditentukan. Setelah menerima *input*, mesin kemudian akan memberikan *output* berdasarkan kumpulan data yang dilabeli. Metode ini cocok digunakan untuk penyelesaian masalah *classification* dan *regression* (Gunova, 2021).

Algoritma yang termasuk ke dalam teknik supervised *learning* di antaranya Decision Tree, K-Nearest Neighboor (KNN), Naive Bayes, Regresi, dan Super Vector *Machine* (Pamungkas, Prasetya, & Kharisudin, 2020).



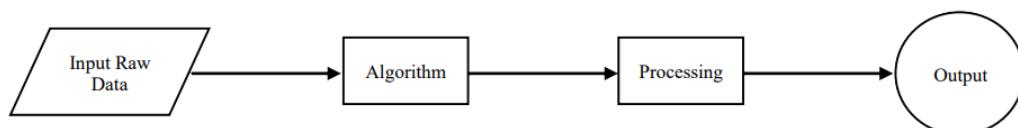
(Sumber : Gunova, 2021)

**Gambar 2.3. Cara Kerja Supervised Learning (SL)**

#### 2.4.1.2. Unsupervised learning

Unsupervised *Learning* (UL) merupakan sebuah metode ML di mana metode ini hanya memberikan data *input* saja ke dalam mesin. Setelah menerima *input*, mesin kemudian akan memberikan *output* berdasarkan pola data *input* yang diterima. Metode ini cocok digunakan untuk penyelesaian masalah pengelompokan data, baik itu association maupun clustering (Gunova, 2021).

Beberapa Algoritma dalam unsupervised *learning* di antaranya DBSCAN, Fuzzy C-Means, K-Means, dan Self Organizing Map. DBSCAN pengelompokan berdasarkan kepadatan (density) data, konsep kepadatan menghasilkan status dari data yaitu core (inti), border (batas), dan noise (Ashari, Otniel, & Rianto, 2019).

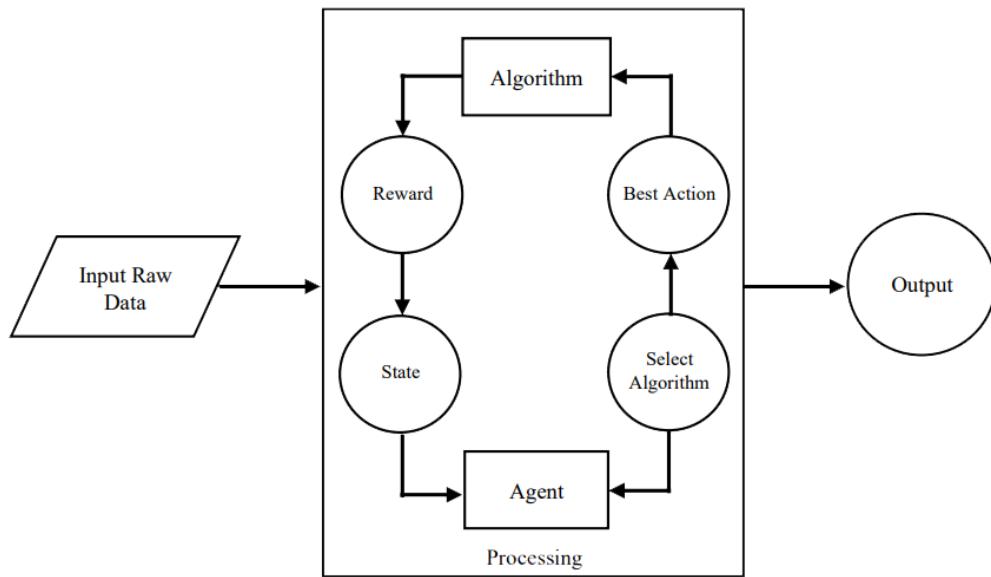


(Sumber : Gunova, 2021)

**Gambar 2.4. Cara Kerja Unsupervised Learning (UL)**

#### 2.4.1.3. Reinforcement learning

Reinforcement *machine learning* adalah algoritma yang mempunyai kemampuan untuk berinteraksi dengan proses belajar yang dilakukan, algoritma ini akan memberikan poin (reward) saat model yang diberikan semakin baik atau mengurangi poin (error) saat model yang dihasilkan semakin buruk. Salah satu penerapan yang sering dijumpai yaitu pada mesin pencari (Fajarsari, 2020).

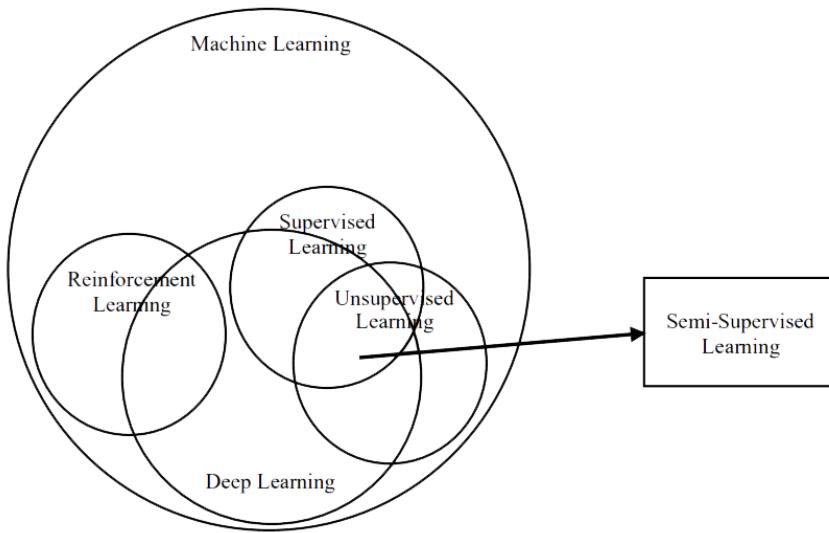


(Sumber : Gunova, 2021)

**Gambar 2.5. Cara Kerja Reinforcement Learning (RL)**

### **2.5. Deep Learning**

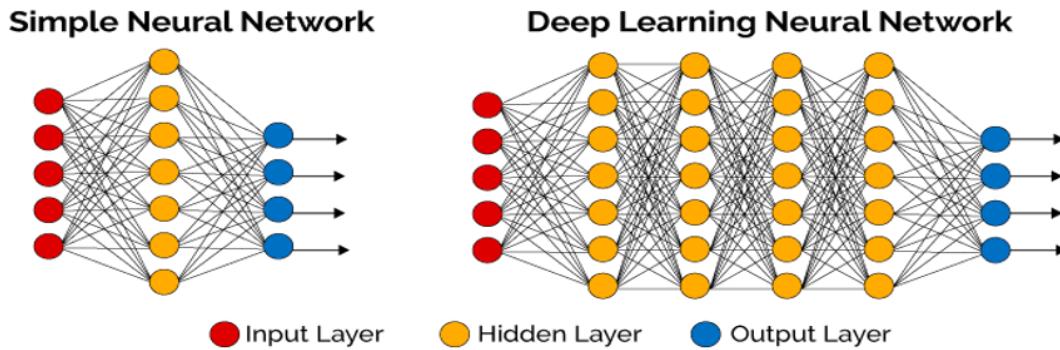
Sejak tahun 2006, *Deep Structured Learning* atau yang lebih dikenal dengan *Deep Learning* atau *Hierarchical Learning* telah muncul sebagai area baru dalam penelitian *Machine Learning* yang berdasarkan pada suatu set algoritma yang mencoba untuk memodelkan abstraksi tingkat tinggi pada data dengan menggunakan graf yang mendalam dengan beberapa *layer* pengolahan, yang terdiri dari beberapa transformasi linier dan non-linier (Francois, 2018).



(Sumber : Gunova, 2021)

**Gambar 2.6. Salah Satu Hubungan antar metode yang dalam ML (UL, SL, RL, dan DL)**

*Deep Learning* sendiri adalah cabang ilmu *machine learning* berbasis *Neural Network (NN)* atau bisa dikatakan sebagai perkembangan dari *Neural Network* (Ilahiyah & Nilogiri, 2018). Metode pendekatan *Deep Learning* mengklasifikasi data dalam dua sesi yaitu sesi *training* dan *testing*. Pada sesi *training* mempelajari ekstraksi fitur dari setiap data supaya bisa membedakan satu *label* dengan *label* yang lain. Pada sesi *testing* data-data yang diuji dapat di analisa dari hasil sesi *training* (Azizah, Umayah, & Fajar, 2018).



(Sumber : Savalia & Emamian, 2018)

**Gambar 2.7. Perbedaan Simple Neural Network dan Deep Learning**

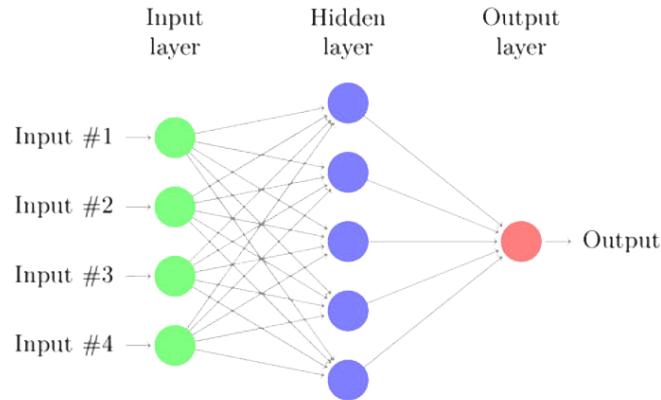
Dalam *Deep Learning*, sebuah komputer belajar mengklasifikasi secara langsung dari gambar atau suara. Metode *Deep Learning* menggunakan CPU dan RAM dalam proses komputasi, dan juga memanfaatkan GPU sehingga proses komputasi data yang besar dapat berlangsung lebih cepat (Ilahiyah & Nilogiri, 2018).

## 2.6. Jaringan Syaraf Tiruan

*Jaringan saraf tiruan (JST) / Artificial Neural Network (ANN) / Neural Network (NN)* adalah jaringan komputasi terinspirasi dari cara kerja otak manusia karena terbukti Otak manusia melakukan sesuatu hal yang sama persis seperti, hierarki pertama pada *neuron* menerima *formasi* pada visual cortex yang sensitif terhadap gambaran tepi dan gumpalan khusus (Fran ois, 2018).

Neural Network mempunyai lapisan masukan (*input layer*) dan lapisan keluaran (*output layer*). Pada setiap lapisan mempunyai satu atau beberapa *unit neuron*, dan mempunyai sebuah fungsi aktivasi dari *unit* tersebut untuk menentukan sebuah keluaran. Untuk meningkatkan kemampuan dari NN, dapat ditambahkan lapis tersembunyi atau *hidden layer*. Data *training* dapat digunakan untuk melatih NN,

semakin banyak data *training* maka akan semakin baik unjuk kerja dari NN tersebut. Tetapi NN juga mempunyai keterbatasan pada jumlah lapisan, karena semakin banyak jumlah lapisan semakin banyaknya juga jumlah iterasi atau *training* yang dibutuhkan (Priyanto, Zarlis, Mawengkang, & Efendi, 2019).



(Sumber : Shekar, S'a, Ferreira, & Soares, 2018)

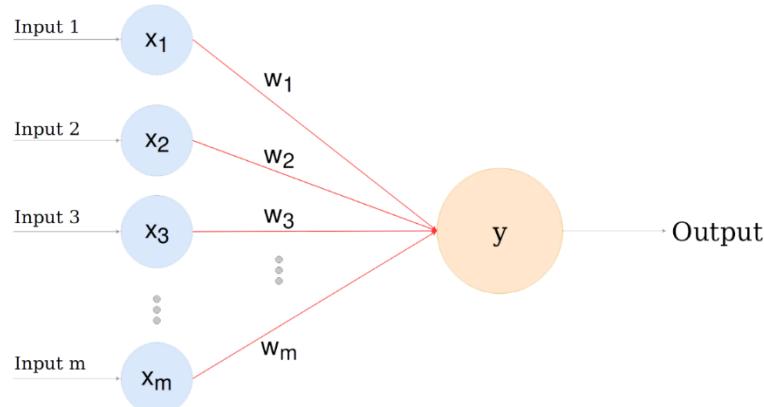
**Gambar 2.8. Skema Jaringan Saraf Tiruan**

Seperti yang terlihat pada Gambar 2.8 Lapisan-lapisan penyusun JST tersebut dapat dibagi menjadi 3, yaitu (Satria, 2018) :

1. Lapisan *Input, unit-unit* di dalam lapisan *input* disebut *unit-unit input*. *Unit-unit input* tersebut menerima pola *inputan* data dari luar yang menggambarkan suatu permasalahan (Satria, 2018)
2. Lapisan tersembunyi, *unit -unit* di dalam lapisan tersembunyi disebut *unit-unit tersembunyi* (Satria, 2018)
3. Lapisan *output, unit-unit* di dalam lapisan *output* disebut *unit-unit output* (Satria, 2018)

### 2.6.1. Arsitektur Jaringan Syaraf Tiruan

Arsitektur atau struktur neural network adalah gambaran susunan komponen *layer* dan *neuron* pada *input*, *hidden* dan *output* yang terhubung dengan *weight* atau *weight*, activation *Function* dan *learning Function*. Perceptron dan Multilayer Perceptron adalah dasar dari jaringan saraf tiruan. Sebuah perceptron adalah algoritma klasifikasi biner yang dimodelkan setelah berfungsinya otak manusia, hal ini dimaksudkan untuk meniru *neuron*. Meskipun perceptron memiliki struktur sederhana tetapi memiliki kemampuan untuk belajar dan menyelesaikan masalah yang sangat kompleks. Neural Network yang paling populer adalah jaringan multi perceptron *feed-forward* yang dilatih melalui algoritma *backpropagation* (Ranjit, Shrestha, Subedi, & Shakya, 2018).



(Sumber : cllau, 2020)

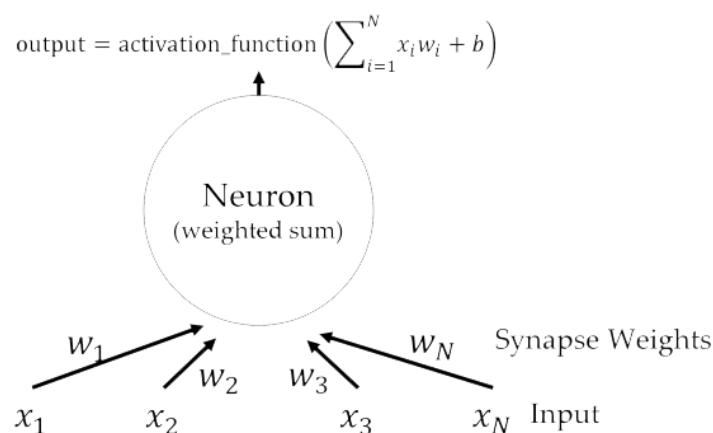
**Gambar 2.9. Perceptron *Input* dan *Output***

Menurut Revi, Solikhun, & Safii JST memiliki beberapa arsitektur jaringan yang sering digunakan dalam berbagai sistem. Arsitektur JST tersebut, antara lain sebagai berikut (Revi, Solikhun, & Safii, 2018) :

### 2.6.1.1. Single Perceptron

ANN yang bentuknya paling kecil disebut single perceptron yang hanya terdiri dari sebuah *neuron*, seperti terlihat pada Gambar 2.9 (Priyanto, Zarlis, Mawengkang, & Efendi, 2019).

Single perceptron hanya terdiri dari 1 lapisan *input* dan 1 lapisan *output*. Setiap *neuron* yang terdapat di dalam lapisan *input* selalu terhubung dengan setiap *neuron* yang terdapat pada lapisan *output*. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi (Revi, Solikhun, & Safii, 2018).



(Sumber : Putra J. G., 2020)

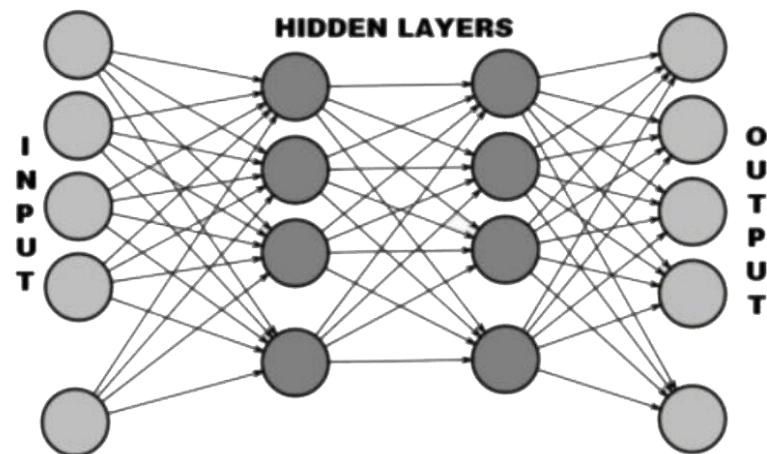
**Gambar 2.10. Single Perceptron**

Secara matematis pada Gambar 2.10 terdapat *feature vector*  $x$  yang menjadi *input* bagi *neuron* tersebut. *Feature vector* merepresentasikan suatu data point, event atau instance. *Neuron* akan memproses *input*  $x$  melalui perhitungan jumlah perkalian antara nilai *input* dan *synapse weight*, yang dilewatkan pada fungsi non-linear. Pada *training*, yang dioptimasi adalah nilai *synapse weight* (*learning*

*parameter*). Selain itu, terdapat juga *bias b* sebagai kontrol tambahan. *Output* dari *neuron* adalah hasil fungsi aktivasi dari perhitungan jumlah perkalian antara nilai *input* dan *synapse weight*. Ada beberapa macam fungsi aktivasi, misal *step Function*, *sign Function*, *rectifier* dan *sigmoid Function* (Priyanto, Zarlis, Mawengkang, & Efendi, 2019).

### 2.6.1.2. Multilayer Perceptron (MLP)

Multilayer merupakan bentuk lapisan perceptron yang digabungkan, dengan menambahkan lebih banyak *layer* dan *neuron* tiap *layer*. Multilayer Perceptron (MLP) sendiri merupakan arsitektur yang paling banyak digunakan untuk jaringan saraf (Sen, Sugiarto, & Rochman, 2020).



(Sumber : Sen, Sugiarto, & Rochman, 2020)

**Gambar 2.11. Deep Neural Network Multilayer Perceptron**

Multilayer Perceptron memiliki ciri khas tertentu yaitu memiliki 3 jenis lapisan yakni lapisan *input*, lapisan *output*, dan lapisan tersembunyi. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih kompleks

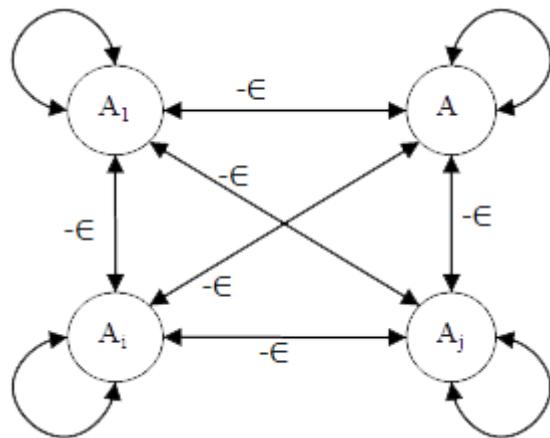
dibandingkan jaringan dengan lapisan tunggal. Namun, proses pelatihan sering membutuhkan waktu yang cenderung lama (Revi, Solikhun, & Safii, 2018).

MLP dilatih dari data pelatihan melalui proses yang disebut *backpropagation*. Proses ini dapat digambarkan sebagai cara untuk memperbaiki kesalahan secara progresif segera setelah terdeteksi. Pada awalnya, semua *weight* ditetapkan secara acak. Kemudian jaringan diaktifkan untuk setiap *input* dalam set pelatihan: nilai disebarluaskan ke depan (*forward propagation*) dari tahap *input* melalui tahap tersembunyi ke tahap *output* di mana prediksi dibuat (Sen, Sugiarto, & Rochman, 2020).

Karena nilai real yang diamati dalam set pelatihan diketahui, maka memungkinkan untuk menghitung kesalahan yang dibuat dalam prediksi. Proses kerja utama dalam backtracking adalah melakukan alur kembali dari *output* menuju *input* dengan menggunakan algoritma pengoptimalan yang tepat, seperti gradient descent, untuk menyesuaikan *weight* (*weight*) jaringan saraf dengan tujuan mengurangi kesalahan (Sen, Sugiarto, & Rochman, 2020).

#### **2.6.1.3. Competitive Layer Net**

Bentuk lapisan kompetitif merupakan jaringan saraf tiruan yang sangat besar. Interkoneksi antar *neuron* pada lapisan ini tidak ditunjukkan pada arsitektur seperti jaringan yang lain. Pada jaringan ini sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif atau sering pula disebut dengan prinsip *winner takes all* atau yang menanglah yang mengambil semua bagiannya (Sadli, 2018).



(Sumber : Sadli, 2018)

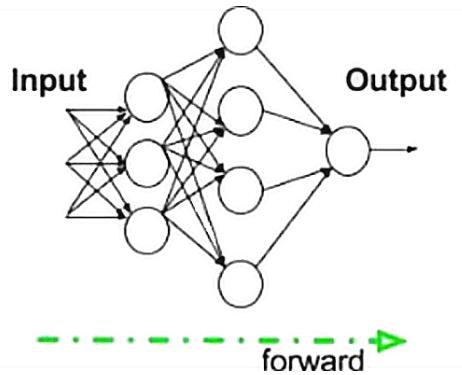
**Gambar 2.12. Competitive Layer Net**

### 2.6.2. Teknik Training Jaringan Syaraf Tiruan

Proses Jaringan Syaraf Tiruan proses *training* terbagi menjadi dua yaitu *Forward Pass* dan *Backpropagation* (Winoto, 2020). Antara lain sebagai berikut :

#### 2.6.2.1. *Forward Propagation / Forward pass*

Pemrosesan dari *layer input* ke *hidden layer* dan kemudian ke lapisan *output* adalah disebut *forward propagation* (Sen, Sugiarto, & Rochman, 2020). *Forward propagation* adalah proses mengolah sinyal *input* dengan *weight* yang tersedia pada saat melewati *Hidden Layer* hingga sampai ke *Output Layer*. Setiap *Layer* memiliki fungsi aktivasi yang berfungsi untuk mengaktifkan atau tidak suatu sinyal (Winoto, 2020).



(Sumber : Sen, Sugiarto, & Rochman, 2020)

**Gambar 2.13. *Forward Propagation***

#### **2.6.2.1.1. Algoritma Training Forward Propagation**

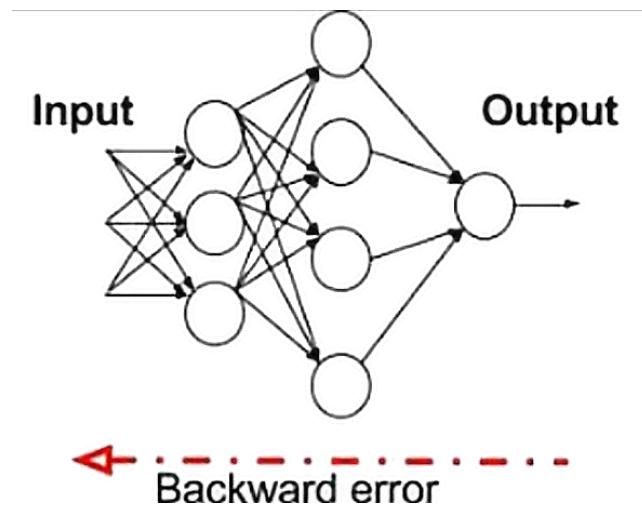
Algoritma pembelajaran untuk JST *forward* propagation adalah sebagai berikut (Azise, Andono, & Pramunendar, 2019) :

1. Masing-masing *input*  $x_i, i = 1, 2, 3, \dots, n$  menerima sinyal  $x_i$ , kemudian melanjutkan ke semua *unit* pada *hidden layer*
2. Pada masing-masing *hidden layer*, menjumlahkan *weight* sinyal *input* dengan persamaan :  $z_{in_j} = v_{0j} + \sum_{i=1}^n x_i v_{ij}$  Untuk menghitung sinyal *output* pada fungsi aktivasi menggunakan persamaan  $z_i = f(z_{in_j})$
3. Masing-masing *unit output*  $y_k, k = 1, 2, \dots, m$  menjumlahkan *weight* sinyal *input* dengan persamaan  $y_{ink} = w_{0k} + \sum_{i=1}^p z_i w_{jk}$  Untuk menghitung nilai *output* dengan menerapkan fungsi aktivasi menggunakan persamaan  $y_k = f(z_{in_j})$

#### **2.6.2.2. Backward propagation / Backward Pass**

*Backpropagation* merupakan metode pelatihan dari Artificial Neural Network yang menggunakan arsitektur multilayer dengan algoritma pembelajaran supervised. Metode ini bertujuan untuk melatih jaringan untuk mendapatkan keseimbangan

antara kemampuan jaringan untuk mengenali pola dalam pelatihan dan memberikan respons yang benar terhadap pola *input* yang hampir sama dengan pola yang dipakai selama pelatihan. Metode ini juga melakukan dua tahapan, yaitu *feedforward* atau perhitungan maju dan *backward* propagation atau perhitungan mundur (Pandji, Indwiarti, & Rohmawati, 2019).



(Sumber : Sen, Sugiarto, & Rochman, 2020)

**Gambar 2.14. Backward Propagation**

#### **2.6.2.2.1. Tahap Training Backward Propagation**

Secara garis besar, *training* jaringan dengan dengan metode *backpropagation* meliputi tiga tahap (Satria, 2018) :

##### **2.6.2.2.1.1. Tahap maju**

Tahap *feedforward* yang dimaksud adalah proses pengolahan *input* dari pola *input training* pada *input layer* sampai respons yang dihasilkan mencapai *output layer* (Satria, 2018).

#### *2.6.2.2.1.2. Tahap perhitungan error propagasi balik*

Respons yang dihasilkan pada *output layer* akan dibandingkan dengan *output target*, kemudian dihitung *errornya*. Bila kriteria untuk kondisi berhenti (stopping condition) belum terpenuhi, maka dilanjutkan ke tahap ketiga (adjustment of the *weights* and *biases*). Namun jika kondisi berhenti sudah terpenuhi, maka proses perhitungan berhenti (Satria, 2018).

#### *2.6.2.2.1.3. Tahap pembaharuan weight dan bias*

Kondisi ini terjadi jika *output* yang diharapkan tidak sesuai, maka jaringan akan bergerak mundur (*backward*) dari *output layer* menuju ke *input layer* dan akan melakukan update *weight* dan bisa serta mengulangi proses dari tahap 1. *Backpropagation* merupakan algoritma pembelajaran yang biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah *weight-weight* yang terhubung dengan *neuron-neuron* yang ada pada lapisan tersembunyinya (Satria, 2018).

#### **2.6.2.2. Algoritma Training Backward Propagation**

Algoritma pembelajaran untuk JST *Backpropagation* adalah sebagai berikut (Mufligh, Sunardi, & Yudhana, 2019) :

1. Inisiasi *weight* (tetapkan dengan nilai acak kecil)
2. Selama syarat kondisi false kerjakan langkah 2-9
3. Untuk setiap pasangan yang akan dilakukan pembelajaran, kerjakan langkah 3-8
4. Setiap *unit input*  $x_i, i = 1, 2, 3, \dots, n$  menerima sinyal dan meneruskan sinyal ke semua *unit* pada lapisan tersembunyi

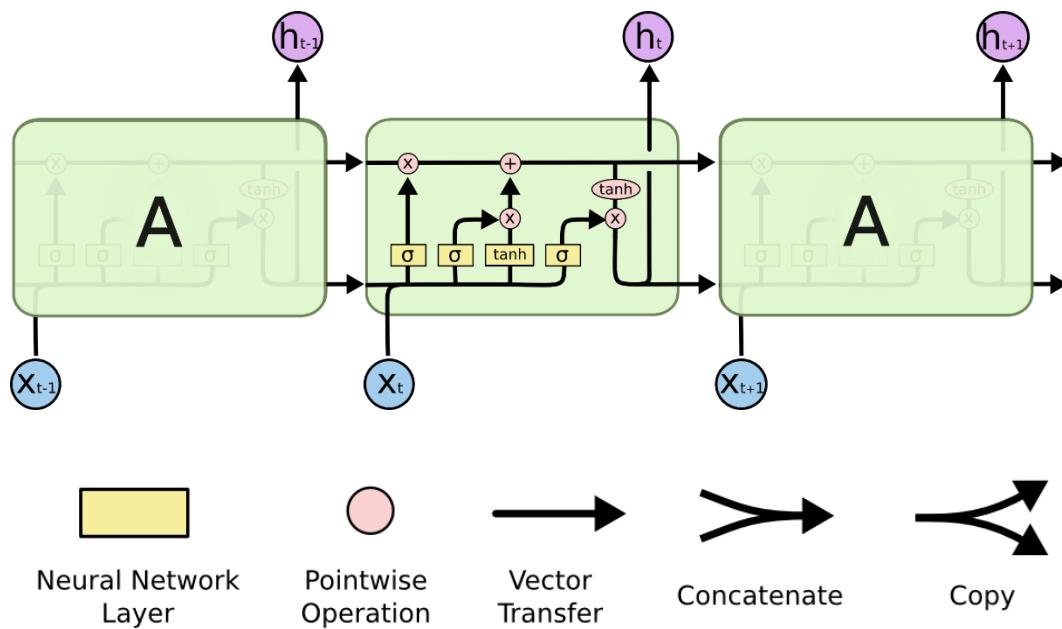
5. Setiap *unit* tersembunyi  $z_j, j = 1, 2, \dots, p$  menjumlahkan *weight* sinyal *input* dengan persamaan berikut:  $z\_in_j = v_{oj} + \sum_{i=1}^n x_i v_{ij}$ , Hitung sinyal *output* dengan fungsi aktivasinya  $z_j = f(z\_in_j)$  Kirimkan sinyal ini ke semua *unit* pada lapisan *output*.
6. Setiap *unit output*  $y_k, k = 1, 2, \dots, m$  menjumlahkan *input* ter*weightnya*  $z\_in_j = v_{oj} + \sum_{i=1}^n x_i v_{ij}$  Hitung sinyal *output* dengan fungsi aktivasinya  $y_k = (t_k - y_k)f'(y\_in_k)$
7. Setiap *unit output*  $y_k, k = 1, \dots, m$  menerima pola target sesuai dengan pola *input* pelatihan, kemudian hitung *error* seperti persamaan  $\delta_k = (t_k - y_k)f'(y\_in_k)$  Hitung suku koreksi *weight* (digunakan untuk perbaruan wjk)  $\Delta w_{jk} = \alpha \delta_k z_j$  Hitung suku koreksi *bias* (digunakan untuk perbaruan wok)  $\Delta w_{ok} = \delta \alpha_k$  Kirimkan  $\delta_k$  ke *unit-unit* dilapis bawahnya
8. Setiap *unit* tersembunyi  $z_j, j = 1, \dots, p$  menjumlahkan delta *inputnya* (dari *unit-unit* yang berada pada lapis diatasnya)  $\delta\_in_j = \sum_{k=1}^m \delta_k w_{jk}$  Hitung suku informasi *error*  $\delta_j = \delta\_in_j f'(z\_in_j)$  Hitung suku koreksi *weight* (untuk perbaruan vij)  $\Delta v_{ij} = \alpha \delta_j x_i$  Hitung suku koreksi *bias* (untuk perbaruan voj)  $\Delta v_{oj} = \alpha \delta_j$  Perbaruan *weight* dan *bias*
9. Setiap *unit output*  $y_k, k = 1, \dots, m$  perbarui *weight-weight* dan *biasnya* :  $w_{jk}(\text{baru}) = v_{ij}(\text{lama}) + \Delta w_{jk}$
10. Setiap *unit* tersembunyi  $z_j, j = 1, \dots, p$  perbarui *weight-weight* dan *biasnya*  $i = 0, \dots, n: v_{ij}(\text{baru}) = v_{ij}(\text{lama}) + \Delta v_{ij}$
11. Uji syarat berhenti, Fungsi aktivasi *sigmoid*

-  $f(x) = \frac{1}{1+e^{-x}}$

$$- f'(x) = f(x)[1 - f(x)]$$

## 2.7. Long Short-Term Memory

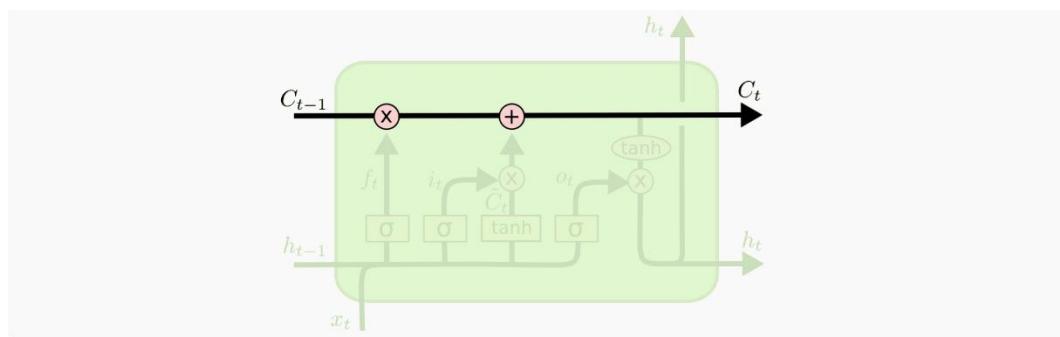
*Long Short-Term Memory (LSTM)* merupakan pengembangan dari *Recurrent Neural Network (RNN)* dengan mengatasi salah satu kekurangan *RNN* yaitu kemampuan pengelolaan informasi dalam periode yang lama yang mana dilakukan modifikasi pada Recurrent Neural Network (RNN) dengan memberi *memory cell* untuk dapat menyimpan informasi dalam waktu yang lama. Diusulkan oleh Sepp Hochreiter dan Jurgen Schmidhuber pada tahun 1997, *LSTM* banyak dipilih untuk prediksi berbasis waktu atau *timeseries* karena dikenal lebih unggul dan handal dalam melakukan prediksi dalam waktu lama dibanding algoritma lain (Zahara, Sugianto, & Ilmiddafiq, 2019; Manaswi, 2018).



(Sumber : Manu, 2021)

**Gambar 2.15. Arsitektur LSTM berisi empat layer yang saling berinteraksi.**

Kunci LSTM adalah *cell state*, garis horizontal yang melewati bagian atas diagram keadaan sel seperti ban berjalan. Ini berjalan lurus ke bawah seluruh rantai, memiliki beberapa linier kecil interaksi (Ghosh, Bose, Maji, Debnath, & Sen, 2019). Untuk setiap sel memori memiliki tiga *layer sigmoid* dan satu *layer tanh* (Qiu, Wang, & Zhou, 2019).



(Sumber : Manu, 2021)

**Gambar 2.16. Alur Informasi *Cell state* pada LSTM**

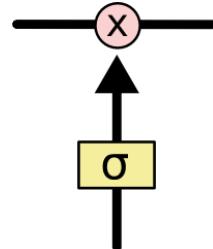
Keterangan (Pardede & Ibrahim, 2020) :

$C_t$  : *Cell state*

$C_{t-1}$  : Nilai *Cell state* sebelum order ke  $t$

Pada Gambar 2.16, garis horizontal yang melalui bagian atas diagram dikenal sebagai *cell state* ( $C_{t-1}, C_t$ ). Ini bertindak seperti ban berjalan yang berjalan di seluruh jaringan. Ini membawa informasi dari sel sebelumnya ke saat ini dan seterusnya (Hiransha, Gopalakrishnan, Menon, & Soman, 2018). Kemampuan untuk menambah atau menghapus informasi ke *cell state* dikendalikan oleh struktur yang disebut *gate*. *Gate* digunakan untuk secara opsional membiarkan informasi lewat. Informasi yang di saring melalui struktur *gate* yang akan mempertahankan

dan memperbarui *cell state* memori (Ghosh, Bose, Maji, Debnath, & Sen, 2019; Qiu, Wang, & Zhou, 2019).



(Sumber : Manu, 2021)

**Gambar 2.17. Layer sigmoid mengeluarkan angka antara nol dan satu.**

Pada Gambar 2.17 *layer sigmoid* mengeluarkan angka antara 0 dan 1, menggambarkan berapa banyak dari setiap komponen yang harus dilewati. Nilai 0 berarti “jangan biarkan apa pun lewat”, sedangkan nilai 1 berarti “biarkan semuanya lewat!” (Ghosh, Bose, Maji, Debnath, & Sen, 2019).

### 2.7.1. Proses *Training* dan *Testing* Pada LSTM

LSTM disebut juga sebagai jaringan saraf dengan arsitektur yang mudah beradaptasi, sehingga bentuknya dapat disesuaikan, tergantung pada aplikasinya. *Long Short-Term Memory* merupakan turunan dari metode RNN (Recurrent Neural Network). Recurrent Neural Network merupakan jaringan saraf berulang yang didesain khusus untuk menghadapi data berurutan (*sequence* data) (Wiranda & Sadikin, 2019).

#### 2.7.1.1. *Training Model LSTM*

Beberapa tahapan dalam proses *training* model LSTM dengan *backpropagation* adalah (Arfan & ETP, 2019) :

1. Inisialisasi *weight* awal
2. *Input* data *training*
3. Perhitungan LSTM pada setiap *input* yaitu dimulai dengan *forget gates*, fungsi *input gates*, fungsi *cell states* dan yang terakhir fungsi *output gates*.
4. Perhitungan standard deviasi RMSE untuk mendapatkan nilai selisih antara nilai LSTM dengan target *output*.
5. Perhitungan gradien untuk menentukan nilai *weight* supaya hasil *loss* mendekati 0 dengan menggunakan *Backpropagation Through Time* (BTTP).
6. Setelah mendapatkan nilai gradien, maka dilanjutkan dengan persamaan fungsi optimasi dan update *weight*.
7. Kembali ke langkah dua sebanyak *epoch* yang telah ditentukan.

Berdasarkan Penjabaran di atas, Pembentukan model LSTM diawali dengan menginisialisasi paramater yang dibutuhkan yaitu *hidden layer* (lapisan tersembunyi), *units* (memori sel), *epoch* (putaran), dan *batch size* (jumlah sampel data). Setelah model dibentuk maka data akan dilatih dengan melewati mekanisme *gates* pada LSTM. Data akan dilatih terus hingga mencapai batas *error* yang diinginkan dengan penentuan serta pengubahan *parameter* yang digunakan (Agusta, Ernawati, & Muliawati, 2021).

Ketika data sudah mencapai target yang diinginkan, proses iterasi akan berhenti dan berikutnya model akan diuji dengan data pengujian atau dapat mengulang kembali proses pelatihan. Proses iterasi ini juga diolah dengan menggunakan fungsi optimasi dan *dropout*. Optimasi berguna untuk menentukan *weight* optimal dan mengurangi kesalahan sehingga dapat memaksimalkan

keakuratan model. Sedangkan *dropout* berguna untuk mencegah terjadinya overfitting pada model (Agusta, Ernawati, & Muliawati, 2021).

### **2.7.1.2. Testing Model LSTM**

Pengujian / *Testing* ini dengan mengambil data *testing* kemudian dibandingkan dengan data yang dihasilkan dengan metode LSTM pada rentang waktu yang ditentukan dengan metode akurasi yang digunakan menggunakan Standard deviasi RMSE (Arfan & ETP, 2019).

### **2.7.2. Fungsi Aktivasi Pada LSTM**

Fungsi aktivasi sangat berperan dalam mengaktifkan setiap *neuron* pada *jaringan saraf tiruan* serta menentukan keluaran dari suatu jaringan saraf tiruan (Susilawati & Muhathir, 2019). Berikut adalah beberapa fungsi aktivasi yang di gunakan dalam penelitian ini :

#### **2.7.2.1. Sigmoid ( $\sigma$ )**

Fungsi aktivasi *sigmoid* merupakan fungsi non-linear. *Input* untuk fungsi aktivasi ini berupa bilangan real dan *output* dari fungsi aktivasi ini memiliki range antara 0 sampai 1 (Suhermi, Suhartono, Dana, & Prastyo, 2018). Berikut ini perhitungan dari fungsi aktivasi *sigmoid* :

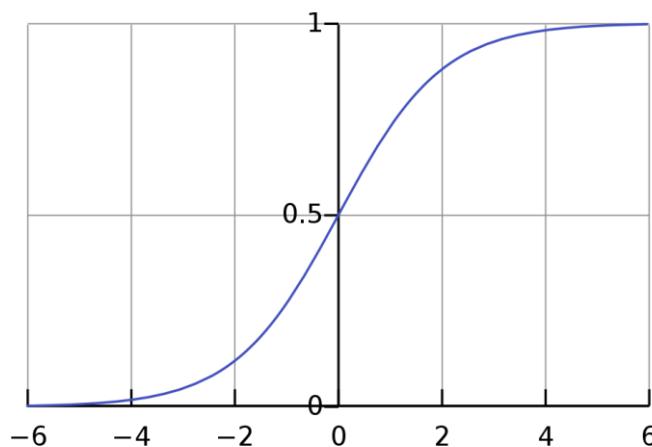
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Di mana :

$x$  : data *input*

$e$  : konstanta matematika (2,718281828...)

Fungsi *sigmoid* mentransformasi range nilai dari *input* x menjadi antara 0 dan 1. Jika *inputnya* sangat negatif, maka keluaran yang didapatkan adalah 0, sedangkan jika *input* sangat positif maka nilai keluaran yang didapatkan adalah 1. Fungsi ini memiliki kekurangan yaitu dapat mematikan *gradient*, ketika aktivasi dari *neuron* mengeluarkan nilai yang berada pada range 0 atau 1, di mana gradient di wilayah ini hampir bernilai 0. Kemudian *output* dari *sigmoid* tidak zero-centered (Suhermi, Suhartono, Dana, & Prastyo, 2018).



(Sumber : Deng, Tong, Lan, & Huang, 2020)

**Gambar 2.18. Illustrasi Sigmoid**

### 2.7.2.2. Hyperbolic (*Tanh*)

Fungsi aktivasi *Tanh* merupakan fungsi non-linear. *Input* untuk fungsi aktivasi ini berupa bilangan real dan *output* dari fungsi tersebut memiliki range antara -1 sampai 1 (Suhermi, Suhartono, Dana, & Prastyo, 2018). Berikut ini perhitungan dari fungsi aktivasi *tanh* :

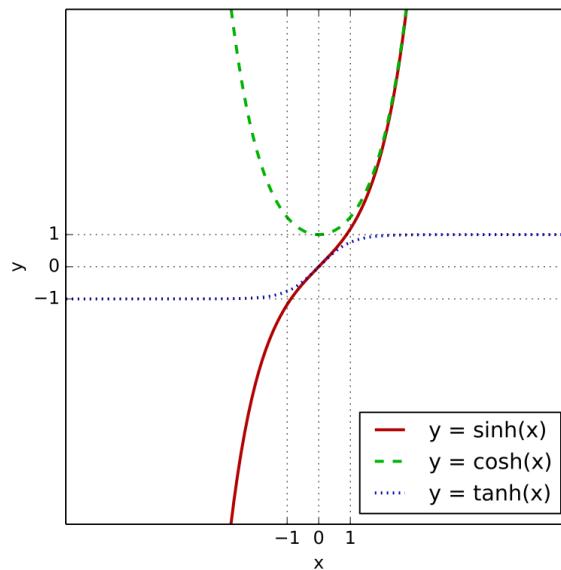
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Di mana :

$x$  : data *input*

$e$  : konstanta matematika ( $2,718281828\dots$ )

Sama seperti fungsi *sigmoid*, fungsi ini memiliki kekurangan yaitu dapat mematikan *gradient*, akan tetapi fungsi ini juga memiliki kelebihan yaitu *output* yang dimiliki fungsi *Tanh* merupakan zero-centered. Dalam pengaplikasiannya fungsi *Tanh* lebih menjadi pilihan jika dibandingkan dengan fungsi *sigmoid*. Fungsi Perlu diketahui fungsi *tanh* merupakan pengembangan dari fungsi *Sigmoid* (Suhermi, Suhartono, Dana, & Prastyo, 2018).



(Sumber : Flywind, 2018)

**Gambar 2.19. Ilustrasi *Tanh***

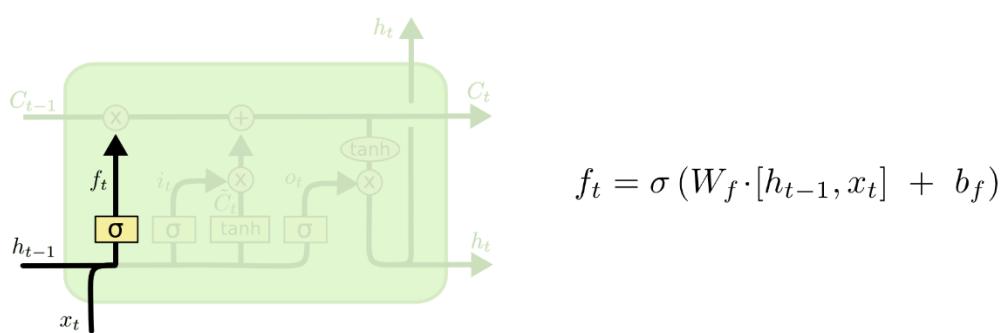
### 2.7.3. Langkah *Training Gates* Pada LSTM

LSTM memiliki tiga di antaranya gerbang, untuk melindungi dan mengontrol *cell state*, Struktur gerbangnya mencakup *forget gate*, *input gate*, dan *output gate*. *Gate* terdiri dari *layer* jaring saraf *sigmoid* dan operasi perkalian pointwise (Ghosh, Bose,

Maji, Debnath, & Sen, 2019; Qiu, Wang, & Zhou, 2019). Berikut ialah keterangan setiap *gate* yang ada pada Gambar 2.15 :

### 2.7.3.1. *Forget Gate*

Pada *forget gate* informasi pada setiap data *input* yang akan diolah dan dipilih data mana saja yang akan disimpan atau dibuang pada *memory cells*. Fungsi aktivasi yang digunakan pada *forget gate* ini adalah fungsi aktivasi *sigmoid*. Di mana hasil keluarannya antara 0 dan 1. Jika keluarannya adalah 1 maka semua data akan disimpan dan sebaliknya jika keluarannya 0 maka semua data akan dibuang (Aldi, Jondri, & Aditsania, 2018). Dengan rumus seperti pada Gambar 2.20 :



(Sumber : Manu, 2021)

**Gambar 2.20. Persamaan *Forget Gate* pada LSTM**

Keterangan (Pardede & Ibrahim, 2020) :

$f_t$  : *Forget gate*

$\sigma$  : Fungsi Aktivasi *Sigmoid*

$W_f$  : Nilai *Weight* untuk *Forget gate*

$h_{t-1}$  : Nilai *output* sebelum order ke  $t$

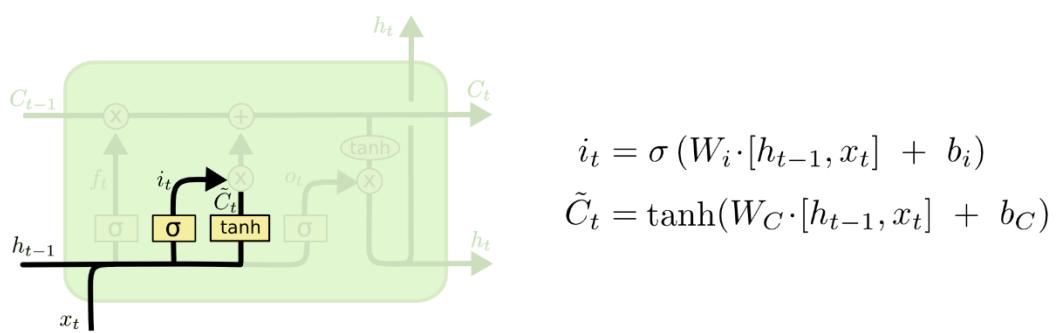
$x_t$  : Nilai *input* pada order ke  $t$

$b_f$  : Nilai bias pada Forget gate

Pada Gambar 2.20 LSTM memutuskan informasi apa yang akan dibuang dari *cell state*. Keputusan ini dibuat oleh *layer sigmoid* yang disebut "layer forget gate." ( $f_t$ ). Terlihat pada  $h_{t-1}$  dan  $x_t$  dan nilai *output* antara angka 0 dan 1 untuk setiap angka dalam *cell state*  $C_{t-1}$  pada Gambar 2.16. *Output* dari 1 mewakili 'sepenuhnya simpan ini' sementara 0 mewakili 'singkirkan ini sepenuhnya' (Boruah & Barman, 2018).

### 2.7.3.2. Input Gate

Pada *input gate* terdapat dua *gates* yang akan dilaksanakan, pertama akan diputuskan nilai mana yang akan diperbarui menggunakan fungsi aktivasi *sigmoid*. Selanjutnya fungsi aktivasi *tanh* akan membuat vektor nilai baru yang akan disimpan pada *memory cell* (Aldi, Jondri, & Aditsania, 2018). Dengan rumus seperti pada Gambar 2.21 :



(Sumber : Manu, 2021)

**Gambar 2.21. Persamaan yang melewati Input Gate pada LSTM**

Keterangan (Pardede & Ibrahim, 2020) :

$i_t$  : Input gate

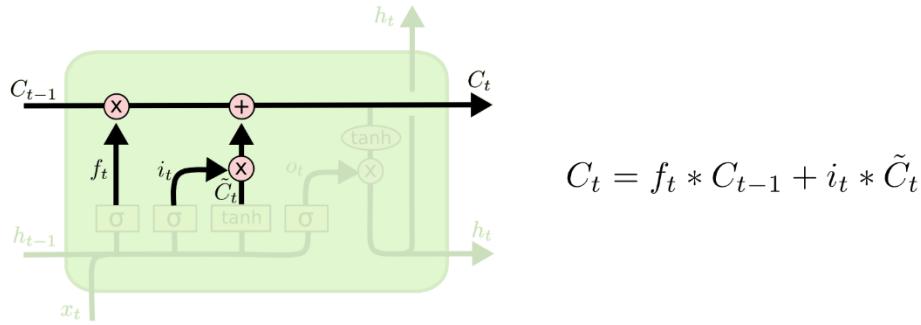
$\tilde{C}_t$	: Nilai baru yang dapat ditambahkan ke <i>cell state</i>
$\sigma$	: Fungsi Aktivasi <i>Sigmoid</i>
$tanh$	: Fungsi Aktivasi <i>Tanh</i>
$W_i$	: Nilai <i>Weight</i> untuk <i>Input gate</i>
$W_c$	: Nilai <i>Weight</i> untuk <i>Cell state</i>
$h_{t-1}$	: Nilai <i>output</i> sebelum order ke $t$
$x_t$	: Nilai <i>input</i> pada order ke $t$
$b_i$	: Nilai <i>bias</i> pada <i>Input gate</i>
$b_c$	: Nilai <i>bias</i> pada <i>cell state</i>

Pada langkah berikutnya di Gambar 2.21 LSTM memutuskan informasi apa yang akan disimpan dari *cell state*. Pertama *layer sigmoid* yang disebut "*layer input gate*" ( $i_t$ ) memutuskan nilai mana yang akan diperbarui. Setelah itu, *layer tanh* membuat vektor nilai kandidat baru,  $\tilde{C}_t$ , yang dapat ditambahkan ke *state* (Ghosh, Bose, Maji, Debnath, & Sen, 2019).

### 2.7.3.3. *Cell state / Memory State*

Pada *cell state gates* akan mengganti nilai pada *memory cell* sebelumnya dengan nilai *memory cell* yang baru. Di mana nilai ini didapatkan dari menggabungkan nilai yang terdapat pada *forget gate* dan *input gate* (Aldi, Jondri, & Aditsania, 2018).

Dengan rumus seperti pada Gambar 2.22 :



(Sumber : Manu, 2021)

**Gambar 2.22. Persamaan Memperbaharui *Cell state* pada LSTM**

Keterangan (Pardede & Ibrahim, 2020) :

$C_t$  : *Cell state*

$f_t$  : *Forget gate*

$C_{t-1}$  : Nilai *Cell state* sebelum order ke  $t$

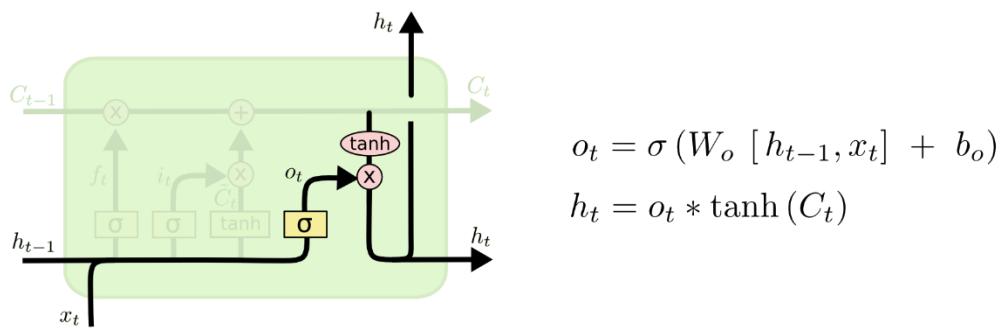
$i_t$  : *Input gate*

$\tilde{C}_t$  : Kandidat konteks baru yang dapat ditambahkan ke *cell state*

Pada langkah selanjutnya di Gambar 2.22, kedua *layer* di gabungkan digabungkan untuk membuat pembaruan ke *cell state*. Pada Langkah inilah nilai *cell state* lama ( $C_{t-1}$ ), akan di perbarui ke nilai dari *cell state* baru ( $C_t$ ) di mana LSTM akan mengalikan *cell state* lama dengan ( $f_t$ ) kemudian ditambahkan dengan ( $i_t * \tilde{C}_t$ ) . Ini adalah nilai kandidat baru, yang diskalakan berdasarkan seberapa banyak memutuskan untuk memperbarui setiap nilai *cell state*. (Ghosh, Bose, Maji, Debnath, & Sen, 2019).

#### 2.7.3.4. Output Gate

Pada *output gate* terdapat dua *gate* yang akan dilaksanakan, pertama akan diputuskan nilai pada bagian *memory cell* mana yang akan dikeluarkan dengan menggunakan fungsi aktivasi *sigmoid*. Selanjutnya akan ditempatkan nilai pada *memory cell* dengan menggunakan fungsi aktivasi *tanh*. Terakhir kedua *gate* tersebut di kalikan sehingga menghasilkan nilai yang akan dikeluarkan (Aldi, Jondri, & Aditsania, 2018). Dengan rumus seperti pada Gambar 2.23 :



(Sumber : Manu, 2021)

**Gambar 2.23. Persamaan melewati *Output Gate* pada LSTM**

Keterangan (Pardede & Ibrahim, 2020) :

$o_t$  : *Output gate*

$C_t$  : *Cell state*

$\sigma$  : Fungsi Aktivasi *Sigmoid*

$\tanh$  : Fungsi Aktivasi *Tanh*

$W_o$  : Nilai *Weight* untuk *Output gate*

$h_{t-1}$  : Nilai *output* sebelum order ke  $t$

$x_t$  : Nilai *input* pada order ke  $t$

$b_o$  : Nilai *bias* pada *Output gate*

$h_t$  : Nilai *output* pada order ke  $t$

Terakhir pada Gambar 2.23 adalah tahap di mana perlu memutuskan apa yang akan hasilkan. *Output* akan didasarkan pada *cell state*, tetapi akan menjadi versi yang difilter. Pertama, peneliti menjalankan *layer sigmoid* yang memutuskan bagian mana dari *cell state* yang akan peneliti hasilkan. Kemudian, *cell state* di tempatkan melalui *tanh* (untuk mendorong nilai menjadi antara -1 dan 1) dan mengalikannya dengan *output gate layer sigmoid*, sehingga hanya akan menampilkan bagian yang putuskan (Ghosh, Bose, Maji, Debnath, & Sen, 2019).

#### 2.7.4. Loss Function

Kinerja pembelajaran diukur dari optimalnya nilai suatu fungsi seperti minimalnya nilai *loss* dan *error*. *Loss* adalah ukuran seberapa dekat atau berbeda model yang dihasilkan dengan data asli, sedangkan *error* merupakan salah satu cara untuk menghitung *loss*. Nilai dari *loss* dan *error* tergantung dari *parameter* pembelajaran yang digunakannya. Kekurangan dari *machine learning* adalah membutuhkan data yang banyak untuk proses pembelajarannya (Putra J. G., 2020). Dalam Penelitian ini fungsi pengukuran nilai *error* yang akan di gunakan adalah *Root Mean Squared Error* (RMSE).

##### 2.7.4.1. Root Mean Squared Error (RMSE)

Tingkat akurasi ditujukan selain secara visual dalam bentuk grafik juga dalam bentuk kuantitatif dengan mengukur nilai RMSE (*Root Mean Square Error*) RMSE berhubungan dengan variasi sebaran frekuensi (*frequency distribution*) dari besar

kesalahan yang diperoleh, tapi tidak dengan variasi kesalahan. (Karno, Hastomo, Nisfiani, & Lukman, 2020). Berikut ini perhitungan dari RMSE :

$$RMSE = \sqrt{\frac{\sum_{t=1}^n (A_t - F_t)^2}{n}}$$

Di mana (Aprian, Azhar, & Nastiti, 2020) :

$n$  : adalah jumlah data

$F_t$  : adalah data prediksi pada waktu ke i

$A_t$  : adalah data asli pada waktu i

Uji Validitas menggunakan *Root Mean Squared Error* (RMSE) dilakukan untuk mengukur hasil akurasi pengujian. RMSE merupakan salah satu contoh *parameter* yang biasa digunakan sebagai indikator untuk mengukur dan membandingkan kemiripan hasil prediksi dan data asli (Aprian, Azhar, & Nastiti, 2020).

### **2.7.5. Batch dan Epochs**

*Epoch* adalah ketika seluruh kumpulan data sudah melalui proses *training* pada *Neural Network* sampai dikembalikan ke awal dalam satu putaran. Dalam *Neural Network* satu *epoch* itu terlalu besar dalam proses pelatihan karena seluruh data diikutkan ke dalam proses *training* sehingga akan membutuhkan waktu cukup lama. Agar mempermudah dan mempercepat proses *training*, biasanya data rate dibagi per *batch* (*Batch Size*).

*Batch size* merupakan jumlah sampel data yang akan disebarluaskan dalam sebuah *neural network*. *Batch size* efisien secara komputasi ketika berhadapan dengan

kumpulan data yang besar. Penentuan nilai dari *batch* size biasanya tergantung peneliti dengan melihat banyak sampel (Thohari & Hertantyo, 2018).

#### **2.7.6. Normalisasi dan Denormalisasi**

Dalam rangka meminimalkan *error* perlu dilakukan normalisasi. Normalisasi berfungsi untuk menghindari terjadinya berbagai anomali data dan tidak konsistensinya data. Normalisasi ini juga bertujuan untuk mengubah ukuran data menjadi lebih kecil tanpa harus mengubah data asli. Teknik normalisasi yang digunakan adalah *minmaxscaling*. Teknik ini digunakan untuk mengatasi perbedaan nilai yang cukup besar antar kumpulan data. Cara kerjanya yakni dengan mengubah nilai pada data aktual menjadi nilai dengan skala (0,1) tanpa mengubah informasi yang ada. Teknik normalisasi dengan *minmaxscaling* memiliki persamaan sebagai berikut (Aldi, Jondri, & Aditsania, 2018).

$$x_{norm} = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$

Di mana :

$x_{norm}$  : Data hasil normalisasi

$x$  : Data asli

$x_{min}$  : Nilai minimum dari data  $x$

$x_{max}$  : Nilai maximum dari data  $x$

Denormalisasi adalah proses pengembalian data hasil normalisasi ke dalam data asli atau data sebenarnya. Hal tersebut dilakukan guna melihat hasil prediksi

dengan cara membandingkan dengan data sebenarnya (Aldi, Jondri, & Aditsania, 2018).

$$x = x_{norm}(x_{max} - x_{min}) + x_{min}$$

Di mana :

$x$  : Hasil *output*

$x_{norm}$  : Nilai dari data normalisasi

$x_{min}$  : Nilai minimal data actual keseluruhan

$x_{max}$  : Nilai maksimal data actual keseluruhan

### 2.7.7. Interpolasi Linear

Interpolasi linier yang sering disebut sebagai interpolasi adalah kemampuan untuk menduga nilai yang terdapat di antara dua nilai lain yang dinyatakan di dalam grafik garis. Interpolasi linier merupakan salah satu metode untuk mengetahui nilai dari suatu interval dua buah titik yang terletak dalam satu garis lurus (Al Amin, Lusiana, & Hartono, 2018).

Interpolasi linier adalah cara mendapatkan nilai di antara dua data berdasarkan persamaan linier. Untuk dapat melakukannya maka minimal harus diketahui dua buah data (Al Amin, Lusiana, & Hartono, 2018).

$$\frac{y - y_a}{y_b - y_a} = \frac{x - x_a}{x_b - x_a}$$

$$x = x_a + (x_b - x_a) \frac{(y - y_a)}{(y_b - y_a)}$$

Di mana (Al Amin, Lusiana, & Hartono, 2018). :

$y$  : Orde data yang akan di interpolasi

$y_a$  : Orde data sebelum data yang akan di interpolasi

$y_b$  : Orde data sesudah data yang akan di interpolasi

$x$  : Data hasil interpolasi

$x_a$  : Data orde sebelum data yang akan di interpolasi

$x_b$  : Data orde sesudah data yang akan di interpolasi

## 2.8. Prediksi / *Forecasting*

Menurut Sucipto & Syaharuddin Prediksi / Peramalan (*forecasting*) adalah kegiatan mengestimasi apa yang akan terjadi pada masa yang akan datang. Peramalan diperlukan karena adanya kesenjangan waktu (*timelag*) antara kesadaran dibutuhkannya suatu kebijakan baru dengan waktu pelaksanaan kebijakan tersebut (Sucipto & Syaharuddin, 2018).

Menurut Putro, Furqon, & Wijoyo prediksi merupakan suatu proses untuk meramalkan atau memperkirakan suatu variabel di masa yang akan datang. Dalam kasus prediksi biasanya data yang sering digunakan adalah data kuantitatif. Prediksi tidak harus menghasilkan suatu jawaban yang pasti kejadian, melainkan berusaha untuk mencari jawaban yang sedekat mungkin dengan kejadian yang akan terjadi (Putro, Furqon, & Wijoyo, 2018).

Secara umum, ada dua jenis prediksi yaitu kualitatif dan kuantitatif. Prediksi kualitatif merupakan prediksi yang bersifat subjektif, hal ini karena didasarkan pada

pengalaman empiris, intuisi pengambilan keputusan dan emosi manusia. Sedangkan, prediksi kuantitatif merupakan prediksi yang bersifat objektif sebab didasarkan pada data aktual dan diolah menggunakan metode tertentu (Surtiningsih, Furqon, & Adinugroho, 2018).

Jenis sesuai yang di jelaskan oleh para ahli sebelumnya prediksi yang digunakan dalam penelitian adalah prediksi kuantitatif. Dikarenakan data yang digunakan adalah data dari masa lalu berupa angka dengan runtutan waktu.

## 2.9. Penelitian Terdahulu

No	Pengarang, Tahun	Metode	Data	Hasil
1	(Moghar & Hamiche, 2020)	<i>Long Short-Term Memory</i>	Dokumentasi saham NYSE (GOOGL dan NKE) dari website yahoo finance	Setelah melatih NN, hasil pengujian peneliti menunjukkan hasil yang berbeda, jumlah <i>epoch</i> serta panjang data memiliki pengaruh yang signifikan terhadap hasil pengujian. Setelah mengamati data peneliti, peneliti dapat melihat bahwa pada awalnya data kurang stabil dan memiliki nilai yang lebih rendah, setelah NKE mulai mengintip nilai yang lebih besar, aset menjadi lebih fluktuatif, maka sifat aset ini beubah. Model peneliti telah kehilangan jejak harga pembukaan sekitar 600 hingga 700 hari pengujian yang sesuai dengan perubahan sifat data.
2	(Karno, Hastomo, Nisfiani, & Lukman, 2020)	<i>Long Short-Term Memory</i> berbasis <i>Gated</i>	Dokumentasi saham bank di Indonesia dari website yahoo finance	Dengan menggunakan paket library seborn di python, hasil perhitungan korelasi dapat ditunjukkan di matrik heatmap dalam bentuk numerik dan tingkatan

		<i>Recurrent Units</i>		warna. Hasil numerik korelasi berupa bilangan dengan rentang -1 dan 1, di mana nilai 1 menunjukkan hubungan yang kuat, dan nilai 0 menunjukkan hubungan yang rendah antara dua data. Jadi dapat dilihat bahwa RMSE mampu meredam perubahan kesalahan yang besar, sebaliknya MSE mampu melihat perubahan kesalahan yang kecil. GRU berupa sel yang berisi hanya 2 <i>gate</i> dengan rangkaian yang lebih sederhana dibandingkan dengan LSTM.
3	(Boruah & Barman, 2018)	<i>Long Short-Term Memory</i>	Identifikasi kalimat pada Novel <i>Bhanumati</i> ditulis oleh Assamese scholar <i>Padmanath Gohainbaruah</i>	<i>Epoch</i> dengan akurasi maksimum untuk model yang berbeda diberikan. Akurasi rata-rata yang terlihat setelah setiap seribu <i>epoch</i> menampilkan kerugian rata-rata setelah setiap seribu <i>epoch</i> dari Uji 1, yang memiliki akurasi maksimum di antara model berbeda yang memiliki konfigurasi berbeda. Tetapi untuk bahasa Assam yang ditranskripsi secara fonetis pada peneliti dapat melihat bahwa akurasi rata-rata meningkat ketika peneliti meningkatkan jumlah <i>neuron</i> dari 128 menjadi 256 di setiap lapisan. Tetapi peningkatan lebih lanjut dari <i>neuron</i> menjadi 512 di setiap lapisan menurunkan akurasi.
4	(Supriyadi, 2019)	<i>Long Short-Term Memory</i>	Observasi sinoptik Stasiun	Metode <i>Deep Learning</i> LSTM digunakan untuk memprediksi <i>parameter</i> cuaca, seperti suhu udara, kelembaban, kecepatan

			Meteorologi Maritim Tanjung Priok	angin, dan tekanan udara. Sedangkan jumlah datanya dibagi dua menjadi <i>training</i> data dan test data dengan rasio 9:1.pada bulan Januari 2019. Diperoleh RMSE <i>parameter</i> suhu udara, kelembaban, kecepatan angin, dan tekanan udara nilainya semakin baik ketika menggunakan <i>Deep Learning</i> LSTM dengan update dibandingkan LSTM tanpa update. Dari <i>parameter</i> cuaca tersebut hanya <i>parameter</i> suhu dan kelembaban udara yang mengalami pertambahan RMSE seiring bertambahnya waktu.
5	(Poornima & Pushpalatha, 2019)	LSTM dengan <i>Weighted Linear Units</i>	Dokumentasi kumpulan data <i>China Meteorological Administration</i>	Dalam penelitian tersebut menyajikan Metode <i>Long Short-Term Memory</i> dengan <i>Weighted Linear Units</i> untuk memprediksi curah hujan. Neural Network dilatih dan diuji menggunakan dataset standar curah hujan. Jaringan yang dilatih akan menghasilkan atribut prediksi curah hujan. <i>Parameter</i> yang dipertimbangkan untuk evaluasi kinerja dan efisiensi model prediksi curah hujan yang diusulkan adalah <i>Root Mean Square Error</i> , akurasi, jumlah <i>epoch</i> , <i>loss</i> , dan <i>learning rate</i> .
6	(Zahara, Sugianto, & Ilmuddafiq, 2019)	<i>Long Short-Term Memory</i>	Dokumentasi secara online dari website <i>Dinas Perdagangan dan Perindustrian</i>	Pembangunan model prediksi dilakukan di lingkungan cloud computing Amazon Web Service tipe EC2. IHK prediksi pada bulan Desember 2016 yaitu 133.98, mempunyai nilai

			<i>Provinsi Jawa Timur</i>	prediksi yang paling mendekati nilai IHK aktual yaitu 133.98 Dari hasil pengujian nilai RMSE tiap algoritma optimasi LSTM yang masih tergolong besar, terlihat bahwa metode LSTM belum bisa disebut metode yang maksimal dalam melakukan prediksi IHK.
7	(Aldi, Jondri, & Aditsania, 2018)	<i>Long Short-Term Memory</i>	Dokumentasi dari website <i>blockchain.info</i>	Pada penelitian tersebut dibangun model LSTM untuk memprediksi harga Bitcoin dengan pengujian <i>parameter</i> komposisi data, jumlah pola <i>timeseries</i> , jumlah <i>hidden neuron</i> dan <i>max epoch</i> . Pada pengujian tersebut didapatkan hasil yang terbaik yaitu dengan komposisi data latih 70% dan data uji 30%, <i>parameter</i> 1 pola <i>timeseries</i> , jumlah 25 <i>neuron hidden</i> , dan <i>max epoch</i> adalah 100 dengan akurasi rata-rata pada data latih 95.36% dan data <i>testing</i> 93.5%.
8	(Putra, Osmond, & Ansori, 2020)	<i>Long Short-Term Memory</i>	Dokumentasi data mentah dari Dinas Perindustrian dan Perdagangan Provinsi Jawa Barat	Ada 4 <i>parameter</i> yang akan diuji untuk setiap komoditas, yang terdiri dari Nilai <i>Epoch</i> , Nilai Lookback, Jumlah <i>Hidden Layer</i> , dan Nilai Train <i>Batch</i> . Jumlah <i>hidden layer</i> adalah banyaknya <i>layer</i> LSTM pada model, terdapat Single- <i>Layer</i> dan Multi- <i>Layer</i> di mana Multi- <i>Layer</i> sendiri terdiri dari 2 dan 3 <i>hidden layer</i> . Tujuan pengujian prediksi di sini yaitu untuk menguji tingkat akurasi dan kecocokan model LSTM dalam memprediksi harga masa depan dengan menggunakan

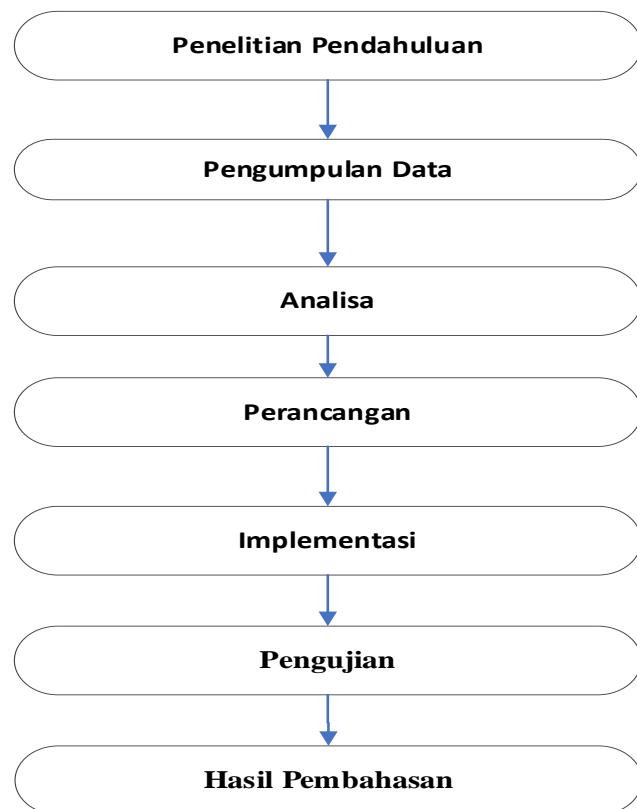
				konfigurasi dari hasil pada pengujian model.
9	(Ghosh, Bose, Maji, Debnath, & Sen, 2019)	<i>Long Short-Term Memory</i>	Dokumentasi dari website resmi <i>Bombay Stock Exchange</i>	Dalam penelitian tersebut, peneliti menganalisis pertumbuhan perusahaan dari berbagai sektor dan mencoba mencari tahu yang merupakan rentang waktu terbaik untuk memprediksi harga saham di masa depan. Prediksi bisa lebih akurat jika model akan berlatih dengan jumlah data yang lebih banyak mengatur. Kerangka kerja ini secara luas membantu dalam analisis pasar dan prediksi pertumbuhan perusahaan yang berbeda dalam rentang waktu yang berbeda.
10	(Rizki, Basuki, & Azhar, 2020)	<i>Long Short-Term Memory</i>	Dokumentasi dari website BMKG	Aplikasi berhasil memproses prediksi curah hujan kota Malang dengan <i>parameter</i> curah hujan. Jumlah <i>neuron hidden layer</i> dengan hasil paling optimal yaitu dengan 256 <i>neuron hidden layer</i> . Jumlah <i>epoch</i> dengan hasil paling optimal yaitu 150 <i>epoch</i> . Komposisi Data Train dan Data Test dengan hasil yang paling optimal yaitu dengan komposisi data train 50% dan data test 50%. Hal ini karena komposisi data train 50% dan data test 50% memiliki tingkat <i>error</i> yang paling rendah yaitu pada data train sebesar 12.079 dan pada data test sebesar 11.288.

## **BAB III**

### **METODOLOGI PENELITIAN**

#### **3.1. Kerangka Penelitian**

Kerangka penelitian merupakan rangkaian kegiatan yang dilakukan dalam suatu penelitian berupa rangkaian grafik yang menggambarkan alur proses penelitian proyeksi curah hujan daerah padang Pariaman menggunakan *Deep Learning* dengan metode *Long Short-Term Memory*, sehingga langkah-langkah yang dilakukan oleh peneliti dalam perancangan ini tidak melenceng dari pokok bahasan dan lebih mudah dipahami, yang diilustrasikan seperti pada Gambar 3.1 berikut.



**Gambar 3.1. Kerangka Penelitian**

### **3.2. Tahapan Penelitian**

Tahapan penelitian merupakan langkah-langkah yang dilakukan untuk mempermudah dalam melakukan penelitian. Tahapan dalam penelitian ini terdiri dari penelitian pendahuluan, pengumpulan data, analisa, perancangan, implementasi, pengujian yang dijelaskan sebagai berikut :

#### **3.2.1. Penelitian Pendahuluan**

Penelitian pendahuluan merupakan proses melakukan pendekatan terhadap objek penelitian. Tujuan dari Penelitian pendahuluan tak lain dapat memberikan solusi terhadap masalah yang di identifikasi. Permasalahan pada proyeksi curah hujan khusus daerah padang Pariaman masih belum pernah di lakukan sama sekali. Oleh karena itu dilakukanlah sebuah penelitian yang nantinya dapat membantu *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* dalam melakukan prediksi curah hujan di daerah padang Pariaman. Penelitian ini nantinya memerikan alternatif lain dalam melakukan prediksi curah hujan di daerah padang Pariaman dengan menggunakan metode *Long Short-Term Memory*.

#### **3.2.2. Pengumpulan Data**

Dalam Penelitian ini semua data bersumber dari *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* dari tahun 1985-2021, dan beberapa pencarian referensi seperti buku-buku, karya-karya ilmiah maupun jurnal, baik yang ada di perpustakaan maupun yang ada di internet yang berhubungan dengan penelitian. Data juga didapat dari studi lapangan dengan melakukan wawancara secara langsung.

### 3.2.2.1. Waktu Penelitian

Penelitian dilakukan dengan merekap data-data pengamatan unsur-unsur cuaca yang dilakukan oleh *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* secara umum yang di lakukan dari bulan November sampai dengan selesai.

**Tabel 3.1. Jadwal Penelitian**

No	Kegiatan	Bulan Ke			
		1	2	3	4
1	Penelitian Pendahuluan	Red	Red	Red	
2	Pengumpulan Data				Green
3	Analisa		Blue	Blue	
4	Perancangan			Purple	Purple
5	Implementasi Sistem			Yellow	Yellow
6	Pengujian Sistem				Orange
7	Pembuatan Laporan			Grey	Grey

### 3.2.2.2. Tempat Penelitian

Objek Penelitian di lakukan di *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* yang beralamat di Jalan Raya Padang – Bukittinggi KM. 51 Kapalo Hilalang Sumatera Barat.

### 3.2.2.3. Metode Penelitian

Metode pada Penelitian ini terdiri dari penelitian perpustakaan, wawancara, penelitian laboratorium. Dengan penjelasan yang di uraikan sebagai berikut :

### **3.2.2.3.1. Penelitian Perpustakaan (Library Research)**

Pada tahap ini peneliti melakukan studi literatur dengan membaca buku, jurnal, makalah dan laporan penelitian yang terkait dengan topik Penelitian. Tujuannya adalah untuk memperkuat permasalahan serta sebagai dasar teori dalam melakukan studi dan juga menjadi dasar untuk melakukan Penelitian terhadap proyeksi curah hujan.

### **3.2.2.3.2. Wawancara (Interview)**

Pada tahap ini peneliti dan responden berhadapan langsung untuk mendapatkan informasi secara lisan dengan tujuan mendapatkan data yang dapat menjelaskan permasalahan yang berhubungan dengan penelitian. Wawancara dilakukan dengan supervisor di *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* terkait dengan permasalahan untuk proses menganalisis masalah, menentukan informasi yang di butuhkah oleh peneliti, seperti pengambilan data, dan lain-lain.

### **3.2.2.3.3. Penelitian Laboratorium (Laboratorium Reaserch)**

Penelitian laboratorium adalah metode penelitian yang di lakukan menggunakan laptop/personal computer (PC). Alat bantu tersebut di gunakan untuk mempraktikkan sekaligus menguji keakuratan metode yang di gunakan.

Adapun Spesifikasi *Hardware* dan *Software* yang di gunakan dalam melakukan penelitian laboratorium ini adalah sebagai berikut :

#### **3.2.2.3.3.1. Perangkat Keras (Hardware)**

- a) Laptop ASUS Model A445LAB
- b) CPU Intel® Core™ i3-5005U CPU @ 2.00GHz (4 CPUs)
- c) *Memory* RAM 12GB

- d) Partisi Penyimpanan Samsung SSD 870 EVO 250GB
- e) GPU Intel(R) HD Graphics 5500

#### *3.2.2.3.3.2. Perangkat Lunak (Software)*

- a) Sistem Operasi Windows 10 Pro 64bit (10.0, Build 19044)
- b) Microsoft Office 365 (64-bit)
- c) Google Chrome (64-bit)
- d) Visual Studio Code (64-bit)
- e) Bahasa Pemrograman Python v.3.9.0 (64-bit)
- f) MySQL Ver 8.0.27 (64-bit)
- g) StarUML Version 4.0.1 (64-bit)
- h) Adobe XD 2021 (64-bit)

### **3.2.3. Analisa**

Analisa dalam penelitian adalah penjabaran dari suatu masalah dari objek yang diteliti yang akhirnya menghasilkan suatu kesimpulan, dalam penelitian ini dimaksudkan untuk mengidentifikasi masalah dalam memprediksi curah hujan berdasarkan data harian klimatologi Dalam proses analisa terdapat tiga tahapan analisa yang harus dilakukan yakni tahapan analisa data, analisa proses dan analisa sistem dengan penjelasan sebagai berikut :

#### **3.2.3.1. Analisa Data**

Analisis data adalah sebuah proses untuk memeriksa, membersihkan, mengubah, dan membuat pemodelan data dengan maksud untuk menemukan informasi yang bermanfaat sehingga dapat memberikan petunjuk bagi peneliti untuk mengambil

keputusan terhadap pertanyaan-pertanyaan penelitian. Data dalam Penelitian ini di dapat langsung dari *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman*.

### **3.2.3.2. Analisa Proses**

Pada tahap analisa proses ini, peneliti melakukan analisa tentang bagaimana proses pemecahan masalah yang di terapkan dan di tentukan sehingga penelitian dapat menghasilkan solusi yang tepat, mulai dari menganalisis masalah, mendokumentasikan, menggunakan metode yang tepat, dan hasil akhir penelitian dalam *format representasi* yang bervariasi dan dapat di mengerti. Dengan begitu di harapkan dapat menjadi solusi yang tepat untuk memecahkan permasalahan proyeksi curah hujan di daerah Padang Pariaman.

### **3.2.3.3. Analisa Sistem**

Selanjutnya pada tahap analisis sistem ini merupakan tahapan yang sangat kritis dan sangat penting, di karenakan kesalahan pada tahapan analisis sistem menyebabkan juga kesalahan ditahap selanjutnya. Tahap analisis sistem merupakan dasar dalam merancang dan merencanakan sistem yang dibuat, analisa sistem dilakukan untuk mengetahui apa saja yang dibutuhkan oleh sistem. Analisa sistem dilakukan untuk mengidentifikasi dan menganalisis keperluan sesuai sistem yang di kembangkan. Agar sistem yang di kembangkan dapat *terorganisir* dengan baik.

### **3.2.4. Perancangan**

Perancangan arsitektur sistem diperlukan agar setiap sistem yang dibangun memiliki konstruksi yang baik, proses pengolahan data yang tepat dan akurat, memiliki nilai, dan memberikan dasar-dasar untuk pengembangan selanjutnya.

### **3.2.4.1. Perancangan Model**

Tahapan perancangan ini, peneliti menggunakan *Unified Modeling Language (UML)* sebagai alat dalam menjelaskan alur perancangan yang dibuat. Tahapan perancangan bertujuan untuk membuat sistem yang dirancang terorganisasi dan terstruktur dengan rancangan sesuai dengan tujuannya, sehingga tidak melenceng dari tujuan Penelitian.

#### **3.2.4.1.1. Use case Diagram**

*Use case Diagram* mendeskripsikan tipe interaksi antara pengguna sistem dengan sistemnya untuk mengetahui fungsi apa saja yang ada di dalam sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

#### **3.2.4.1.2. Class Diagram**

*Class diagram* merupakan suatu diagram yang digunakan untuk menampilkan kelas-kelas berupa paket untuk memenuhi salah satu kebutuhan yang di gunakan nantinya, dan juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku dari sistem.

#### **3.2.4.1.3. Sequence Diagram**

*Sequence Diagram* menjelaskan urutan-urutan kejadian yang terjadi di dalam sistem, serta untuk menggambarkan interaksi antara keseluruhan objek di dalam dan interaksi objek yang disusun dalam sistem berdasarkan urutan waktu dalam bentuk gambaran tahap demi tahap yang seharusnya dilakukan.

#### **3.2.4.1.4. Statechart Diagram**

*Statechart Diagram* digunakan untuk menggambarkan perubahan status yang terjadi dengan menggambarkan transisi serta perubahan pada suatu objek pada sistem selama sistem dijalankan sampai dengan selesai.

#### **3.2.4.1.5. Activity Diagram**

*Activity Diagram* merupakan sebuah diagram yang dapat memodelkan berbagai proses yang terjadi pada sistem yang berfokus pada aktivitas yang terjadi yang terkait dalam suatu proses tunggal. Dengan kata lain, diagram ini menunjukkan bagaimana aktivitas yang dilakukan oleh pengguna.

#### **3.2.4.1.6. Collaboration Diagram**

*Collaboration diagram* dikenal dengan beberapa nama, seperti communication diagram dan interaction diagram, yang mana penggambaran interaksi dan hubungan antara objek dalam sistem.

#### **3.2.4.1.7. Deployment Diagram**

*Deployment diagram* adalah diagram yang digunakan memetakan *software* ke processing node. Menunjukkan konfigurasi elemen pemroses pada saat runtime dan *software* yang ada di dalamnya, dan digunakan untuk menggambarkan detail bagaimana komponen disusun di infrastruktur sistem.

#### **3.2.4.2. Perancangan Interface**

*Interface* merupakan mekanisme komunikasi antara pengguna dengan sistem. Meliputi tampilan layar yang menyediakan navigasi di dalam sistem, layar dan formulir untuk menangkap data, dan laporan yang dihasilkan oleh sistem.

Perancangan *interface* dibuat untuk memberikan penjelasan tentang tampilan yang dihadapkan pada pengguna saat menggunakan sistem untuk membantu mengarahkan alur penelitian masalah sampai ditemukan suatu solusi.

### **3.2.5. Implementasi**

Implementasi sistem merupakan tahap meletakkan sistem sehingga siap untuk dioperasikan. Implementasi bertujuan untuk mengkonfirmasi modul-modul perancangan, sehingga pengguna dapat memberi masukan kepada pengembangan sistem.

Untuk mengimplementasikan sistem yang telah dirancang, maka diperlukan sebuah alat bantu komputer untuk mengoperasikan komputer itu sendiri yang memerlukan tiga buah komponen pendukung seperti *Hardware*, *Software*, dan *Brainware*.

#### **3.2.5.1. Perangkat Keras (*Hardware*)**

*Hardware* yang digunakan untuk merancang atau menjalankan program sistem yang telah dibuat dalam satu *unit* komputer yang lengkap dengan CPU, harddisk sebagai media penyimpanan data.

#### **3.2.5.2. Perangkat Lunak (*Software*)**

Untuk menjalankan program sistem yang dirancang harus menggunakan beberapa *software* pendukung.

#### **3.2.5.3. Manusia (*Brainware*)**

*Brainware* merupakan operator yang berfungsi untuk mengoperasikan atau menjalankan program sistem.

### **3.2.6. Pengujian**

Pengujian sistem merupakan tahap melakukan *testing* untuk mengetahui kesalahan dalam Sistem. Pengujian terhadap sistem dilakukan untuk dapat mengetahui Sistem yang dirancang telah berjalan sesuai dengan yang diharapkan. Sehingga memudahkan *admin* mengetahui informasi terkini dari sistem, dan kemudian mengambil tindakan selanjutnya jika dibutuhkan.

#### **3.2.6.1. Pengujian Lokal**

Pengujian Lokal atau LAN, yaitu pengujian yang dilakukan melalui akses jaringan komputer dengan sistem Local Area Network, dengan cara menghubungkan komputer *server* ke komputer *client* dengan media kabel, yang terlebih dahulu dilakukan konfigurasi IP Address pada komputer *server* dan komputer *client*. Sehingga *client* dapat mengakses data pada komputer *server*, sedangkan pada komputer *server* menyediakan informasi yang dibutuhkan oleh *client*.

#### **3.2.6.2. Pengujian Online**

Pembuatan program ini nantinya bisa diakses secara online dengan di hosting di tempat penyedia web service. Selanjutnya melakukan konfigurasi *database* agar terhubung ke web *server*. Setelah semua proses dilakukan maka website bisa diakses oleh *admin* yang ingin menggunakan sistem ini.

#### **3.2.6.3. Pengujian Aplikasi**

Pengujian Aplikasi dengan menggunakan Black Box, yaitu pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak bahwa aplikasi yang dirancang dapat memenuhi

kebutuhan yang mendasari perancangan aplikasi tersebut dan berjalan sesuai dengan diharapkan.

#### **3.2.6.4. Pengujian *Interface***

Pengujian Antarmuka/*Interface*, yaitu pengujian yang dilakukan untuk melihat bagaimana tampilan akhir dari sistem yang telah dirancang dengan pengamatan secara langsung oleh pengguna interaksi secara langsung dengan model *interface* yang disajikan dalam bentuk prototipe. Proses ini dilakukan agar kesalahan dapat terdeteksi dan di ubah secara cepat.

#### **3.2.7. Hasil Pembahasan**

Hasil pembahasan merupakan hasil observasi atau penelitian. Hasil pembahasan merupakan isi dari suatu bagian penting dari suatu penelitian. Hasil pembahasan memiliki sifat objektif atau subjektif. Hasil pembahasan dapat dijelaskan sebagai hasil interpretasi dari hasil penelitian yang telah dianalisis guna menjawab pertanyaan-pertanyaan dalam penelitian.

Penyajian hasil pembahasan dapat dilakukan secara deskriptif, dengan menggunakan tabulasi, tabel atau grafik, atau dengan menggunakan kombinasi dua atau ketiganya sekaligus. Untuk penelitian ini, hasil pembahasan dijelaskan melalui tabel maupun grafik statistik.

## **BAB IV**

### **ANALISA DAN PERANCANGAN**

#### **4.1. Analisa**

Berdasarkan penjelasan yang telah diuraikan pada bab sebelumnya, maka diperlukan suatu proses penganalisan dengan tujuan untuk memberikan hasil yang akurat berdasarkan informasi maupun data yang sudah peneliti dapatkan untuk di terapkan pada perancangan.

##### **4.1.1. Analisa Data**

Analisa Data adalah tahap rentan dalam mengembangkan sebuah sistem yang padu. Di mana analisa data merupakan tahap awal agar sistem yang di rancang memiliki gambaran sesuai pola data yang terbentuk

Data dalam Penelitian ini data di dapat langsung dari *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman*, data yang di gunakan merupakan data harian dari tanggal *1 January 1985 sampai 31 Desember 2021* data tersebut terdiri dari beberapa fitur / variabel seperti yang terlihat pada Tabel 4.1 berikut :

**Tabel 4.1. Fitur Data Klimatologi BMKG Padang Pariaman**

Kode	Keterangan	Satuan
$Tn$	Temperatur minimum	°C
$Tx$	Temperatur maksimum	°C
$Tavg$	Temperatur rata-rata	°C
$RH\_avg$	Kelembapan rata-rata	%
$RR$	Curah hujan	mm

<i>ss</i>	Lamanya penyinaran matahari	<i>jam</i>
<i>ff_x</i>	Kecepatan angin maksimum	<i>m/s</i>
<i>ddd_x</i>	Arah angin saat kecepatan maksimum	°
<i>ff_avg</i>	Kecepatan angin rata-rata	<i>m/s</i>
<i>ddd_car</i>	Arah angin terbanyak	°

Sesuai data Klimatologi yang di tampilkan pada Tabel 4.1 di mana setiap fitur variabel pada data tersebut merupakan variabel in-dependen. Tidak semua fitur data pada Tabel 4.1 di gunakan pada Penelitian ini, dan peneliti hanya menyertakan fitur *RR*.

**Tabel 4.2. Data Stasiun Klimatologi Kelas II Sicincin Padang Pariaman**

Tanggal	<i>RR</i>	<b>Tn</b>	<b>Tx</b>	<b>Tav_g</b>	<b>RH_avg</b>	<b>ss</b>	<b>ff_x</b>	<b>ddd_x</b>	<b>ff_avg</b>	<b>ddd_car</b>
01-01-1985	0			26,5	72	0	0	0	0	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
13-06-2020	2	22,8	9999	25,5	90	1,9	1	120	0	C
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
10-07-2020	0,4	23	32,2	26,8	85	1	4	100	1	C
11-07-2020	8888	22	32,6	25	91	3	3	210	1	C
12-07-2020	15	22	31,2	25,2	88	6,3	6	250	1	C
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
31-12-2021	24,9	24,1	29,6	25,2	94	4,8	2	30	1	C

Pada Tabel 4.2 data klimatologi pada Tanggal 11-07-2020, data yang bernilai 8888 berarti data tidak diukur, dan data Klimatologi pada Tanggal 13-06-2020 yang

bernilai 9999 berarti tidak ada data (tidak dilakukan pengukuran). Nilai tersebut dianggap nilai yang hilang *Nan*.

#### **4.1.1.1. Preprocessing Data**

Dalam Penelitian ini di lakukan *preprocessing* data dengan data yang di gunakan memiliki beberapa nilai yang hilang *missing values* dan juga data bernilai *NaN* dari data yang di teliti. Nilai-nilai yang hilang / *NaN* tersebut ini muncul dari banyak faktor yang berada di luar kendali staf *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman*. di mana nantinya peneliti melakukan cara penanganan dengan beberapa cara sebagai berikut :

##### **4.1.1.1.1. Interpolate *Nan***

*Interpolate Nan* merupakan Teknik *preprocessing* data yang *NaN* atau missing values dengan dengan melakukan interpolasi linear pada data dengan nilai yang hilang / *NaN*, dan data tidak hilang melainkan data di isi dari nilai rentang nilai sebelum dan sesudahnya. Sebagai contoh peneliti mencoba menghitung missing value dari tanggal 14-06-2020 sebagai berikut.

$$\frac{2 - 1}{5 - 1} = \frac{x - 2}{30 - 2}$$

$$x = 2 + (30 - 2) \frac{(2 - 1)}{(5 - 1)}$$

$$x = 2 + 28 \frac{1}{4}$$

$$x = 9$$

Hasil akhir terlihat seperti yang di jabarkan pada Tabel 4.3 dari tanggal 12-06-2020 - 17-06-2020 sebagai berikut.

**Tabel 4.3. Data Klimatologi Sebelum dan Sesudah di Interpolate NaN**

<b>Tanggal</b>	<b>RR</b>	<b>Tanggal</b>	<b>RR</b>
:	:	:	:
12-06-2020	0.0	12-06-2020	0.0
13-06-2020	2.0	13-06-2020	2.0
14-06-2020	NaN	14-06-2020	9.0
15-06-2020	NaN	15-06-2020	16.0
16-06-2020	NaN	16-06-2020	23.0
17-06-2020	30.0	17-06-2020	30.0
:	:	:	:

Seperti yang terlihat pada Tabel 4.3 baris data tanggal 14-06-2020 - 16-06-2020 dimana nilai *NaN* di gantikan dengan nilai rentang dari nilai sebelum dan sesudahnya berdasarkan kedudukan nilai yang di interpolasi dan nilai rentang terdekat.

#### **4.1.1.1.2. Normalisasi MinMaxScaller**

Sebelum data di processing ada baiknya data di normalisasi terlebih dahulu. di mana data yang di proses memiliki nilai rentang yang sama, tidak ada yang terlalu besar maupun terlalu kecil untuk setiap fitur data yang termasuk. Normalisasi data ini berguna agar proses analisis statistik pada data menjadi lebih mudah.

Normalisasi data yang di gunakan adalah normalisasi data *minmax* scaller, *Minmaxscaling* bekerja dengan scaling data dalam rentang tertentu (range nilai minimum hingga nilai maksimum), mengubah data berada pada rentang nilai 0

sampai 1. Sebagai contoh peneliti mencoba menormalisasi nilai dari tanggal 14-06-2020 sebagai berikut.

$$x_{norm} = \frac{(9 - 0)}{(30 - 0)}$$

$$x_{norm} = 0.3$$

Hasil akhir terlihat seperti yang di jabarkan pada Tabel 4.3 dari tanggal 12-06-2020 - 17-06-2020 sebagai berikut.

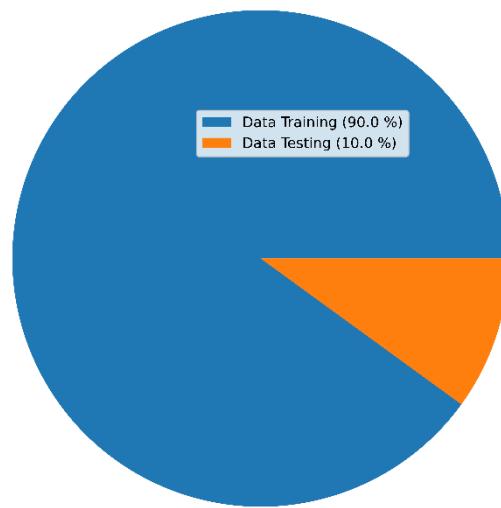
**Tabel 4.4. Data Klimatologi Sebelum dan Sesudah di MinMaxScaller**

Tanggal	RR	Tanggal	RR
⋮	⋮	⋮	⋮
12-06-2020	0.0	12-06-2020	0.0000
13-06-2020	2.0	13-06-2020	0.0667
14-06-2020	9.0	14-06-2020	0.3000
15-06-2020	16.0	15-06-2020	0.5333
16-06-2020	23.0	16-06-2020	0.7667
17-06-2020	30.0	17-06-2020	1.0000
⋮	⋮	⋮	⋮

Berdasarkan pada Tabel 4.4 semua nilai dari kolom *RR* di normalisasi dengan rentang nilai 0 – 1 berdasarkan nilai sebelumnya yang berkorelasi dengan nilai tertinggi dan terendah dari semua baris data yang di sertakan dalam perhitungan minmaxscaler.

#### 4.1.1.2. Pembagian Data *Training* dan *Testing*

Pembagian data *training* di lakukan agar meningkatkan kinerja dari LSTM terhadap data *testing*. Data *training* digunakan untuk proses pelatihan model dengan metode LSTM sehingga terbentuk suatu model yang diuji performansinya terhadap data *testing*. Pembagian data yang digunakan yaitu 90% data *training* dan 10% data *testing*. Jumlah data *training* lebih besar dikarenakan agar mesin pembelajaran lebih terlatih untuk mempelajari model. Sehingga model yang dihasilkan dapat memberikan peramalan data *testing* yang lebih optimal.

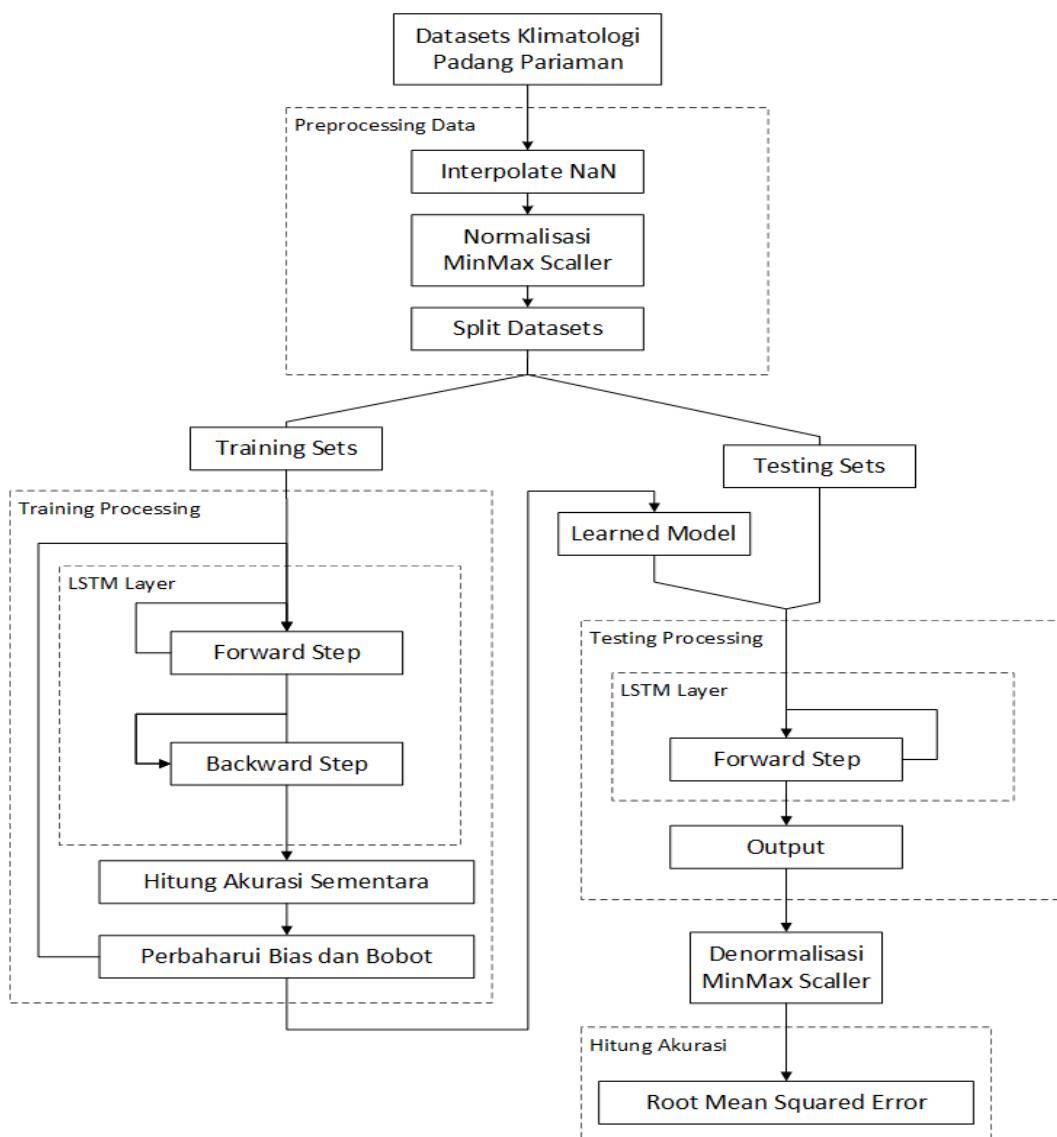


**Gambar 4.1. Skala Pembagian Data *Training* dan Data *Testing***

Pembagian data latih dan data uji seperti yang telihat pada Gambar 4.1 bertujuan agar pembelajaran dapat belajar dari pola yang telah didapatkan dari hasil proses pelatihan yang akan diimplementasikan dalam pengujian data. Proses pelatihan dan pengujian menggunakan metode LSTM akan terus dilakukan hingga didapatkan model yang optimal.

#### 4.1.2. Analisa Proses

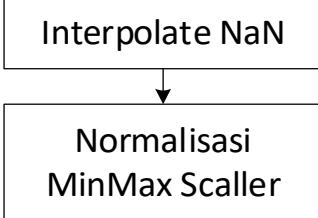
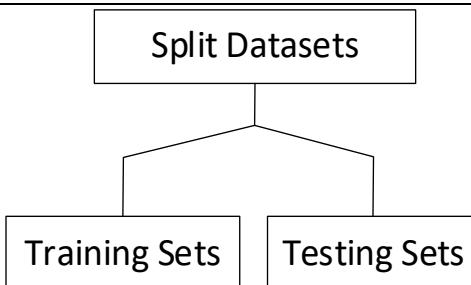
Analisis proses dilakukan untuk mengetahui cara pemecahan masalah sehingga dapat menghasilkan solusi dengan menggunakan metode yang tepat, melalui proses yang kooperatif dan interaktif mulai dari menganalisis masalah, mengidentifikasi masalah, hasil akhir pengamatan dalam berbagai *format* representasi, hingga memeriksa ketepatan pemahaman yang diperoleh. Berikut merupakan flowchart dari langkah Analisa proses :



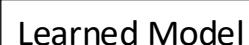
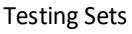
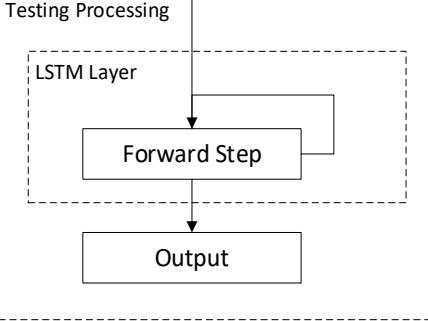
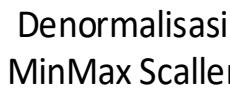
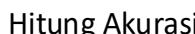
Gambar 4.2. Analisa Proses

Pada Gambar 4.2 terdapat 4 proses utama yaitu terdiri dari *preprocessing* data, *training* processing, *testing* processing, dan hitung akurasi, masing-masing proses harus di lakukan secara bertahap untuk mencapai tujuan penelitian dan mengetahui performansi dari arsitektur yang di rancang terhadap dataset yang di kumpulkan. Untuk penjelasan lebih detail berikut adalah Penjelasan Blok Diagram dari Gambar 4.2 yang terlihat seperti pada Tabel 4.5 di bawah ini :

**Tabel 4.5. Penjelasan Blok Diagram Analisa Proses**

Blok Diagram	Penjelasan
	Kumpulan data / datasets yang di ambil dari <i>Stasiun Klimatologi Kelas II Sicincin Padang Pariaman</i>
	Data yang telah masuk di proses pada tahap <i>preprocessing</i> . Tahapan ini dilakukan agar data yang dimiliki bersih dari NaN value. Penjelasan lengkap dapat di lihat pada <i>Preprocessing Data</i> di atas. Untuk normalisasi data di skalakan dengan interval 0-1.
	Hasil <i>preprocessing</i> data menghasilkan data yang bersih. Data bersih ini dibagi menjadi dua bagian yang itu data pelatihan ( <i>training</i> ) dan data pengujian ( <i>testing</i> ).

<pre> graph TD     TS[Training Sets] --&gt; TP[Training Processing]     subgraph TP [Training Processing]         direction TB         L[LSTM Layer]         FS[Forward Step]         BS[Backward Step]         HAS[Hitung Akurasi Sementara]         PBB[Perbaharui Bias dan Bobot]         L --&gt; FS         FS --&gt; BS         BS --&gt; HAS         HAS --&gt; PBB     end </pre>	<p>Pada <i>training</i> processing terdapat jaringan LSTM dimana proses <i>training</i> menggunakan 2 tahap, yaitu:</p> <ol style="list-style-type: none"> <li>1) <i>Forward Step</i>: proses dimana <i>output</i> value dari sebuah neural network dapatkan sesudah <i>input</i> values dari <i>input-neuron</i> telah di selesaikan di proses. Proses ini di panggil <i>forward</i> karena proses kalkulasinya mulai dari <i>input</i> neural network ke <i>output</i>.</li> <li>2) <i>Backward Step</i>: proses dimana <i>weights</i> neural network di kalkulasikan menggunakan algoritme seperti gradient-descent. Proses ini di panggil <i>backward</i> karena proses kalkulasinya mulai dari <i>output</i> neural network ke <i>input</i>.</li> </ol> <p>Setelah proses <i>training</i> selesai dalam LSTM Layer, nilai akurasi sementara dihitung dengan menggunakan rumus MSE dan RMSE, dan juga <i>parameter bias</i> dan <i>weight</i> di perbaharui dan digunakan pada iterasi dan <i>epoch</i> berikutnya.</p>
---	---

	<p><i>Learned model</i> adalah nilai optimal <i>weight</i> dan <i>bias</i> dari proses <i>training</i> sebanyak <i>epoch</i> yang telah di tentukan yang digunakan untuk melakukan proses pengujian.</p>
  	<p>Hasil optimal dari <i>weight</i> dan <i>bias</i> dari <i>learned model</i> di gunakan untuk melakukan pengujian dari data <i>testing</i> yang berupa <i>input</i> dan <i>label</i>. Pada <i>testing</i> processing terdapat jaringan LSTM dengan proses <i>testing forward step</i> sebanyak data yang di <i>testing</i>. Hasil <i>Output</i> merupakan keluaran dan hasil pengujian dari jaringan LSTM.</p>
	<p>Hasil dari normalisasi dengan interval keluaran 0-1 perlu di lakukan denormalisasi Kembali agar nilai yang terlihat Tambah seperti nyata Kembali.</p>
 	<p>Pada tahapan ini dilihat nilai <i>Root Mean Squared Error</i> pada masing masing pengujian yang dilakukan. Dimana hasil nilai <i>error</i> dari RMSE menjadi acuan ketepatan prediksi.</p>

Seperti yang terlihat pada Tabel 4.5 setelah dilakukan Analisa data, selanjutnya data yang telah melewati tahap *preprocessing* bisa memasuki tahap processing yang mana perlu di persiapkan beberapa kriteria depensi seperti berikut :

1. Memisahkan kumpulan data menjadi 2 bagian yaitu *train set* dan *test set* 90% untuk data *training* 10% untuk data *testing*.
2. Pada pemodelan *Long Short-Term Memory input* data berupa tensor 3D berupa (*ukuran batch, jumlah features, dan timestep/sequence*).
3. Algortima *training* untuk model LSTM di gunakan algoritma pembelajaran *backpropagation*, di mana model pemrosesan berurutan yang terdiri dari dua step, step satu mengambil *input* dalam arah maju (*forward direction*), *input* step kedua ke arah mundur (*backward direction*).
4. Di tentukan nilai *Epoch* sebanyak 30 yang nantinya bisa di ubah sesuai keinginan *admin*.
5. Di tentukan nilai *batch size* sebanyak 64 yang nantinya bisa di ubah sesuai keinginan *admin*.
6. Di tentukan nilai *unit cell* LSTM manjadi 1 yang nantinya bisa di ubah sesuai keinginan *admin*.
7. Di tentukan nilai *sequence length / jumlah timestep* dari data yang di gunakan.
8. Di tentukan nilai *learning rate* di set nilai awalan 0.01 yang nantinya bisa di ubah sesuai keinginan *admin*.
9. Di tentukan nilai probabilitas *dropout* di set nilai awalan 0.1 yang nantinya bisa di ubah sesuai keinginan *admin*, pengecualian apabila jumlah *unit cell* LSTM hanya 1 maka nilai probabilitas *dropout* 0.

10. Fungsi optimasi yang di gunakan adalah *Stochastic Gradient Descent* (SGD).

11. Nilai kesalahan / *error* value dari peramalan digunakan *loss Function* RMSE.

#### **4.1.2.1. Proses *Training* Model**

Pada tahap *training* model LSTM dilakukan secara *backpropagation* sesuai data yang di gunakan. Berikut beberapa tahapan dalam proses *training* model LSTM dengan *backpropagation* sebagai berikut :

1. Menginisialisasi nilai *epoch*, *batch*, *timestep/sequence*, *features*, *dropout*, *units*, dan *learning rate* yang nantinya nilai hyperparamater yang dibutuhkan seperti nilai *weight* awal, *hidden layer* (lapisan tersembunyi) secara otomatis di tentukan oleh sistem.
2. *Input* data *training* berbentuk tensor 3D.
3. Melakukan *training* model LSTM pada setiap *input* yaitu dimulai dengan *forget gates*, *input gates*, *cell states* dan yang terakhir *output gates*.
4. Perhitungan RMSE untuk mendapatkan nilai selisih antara nilai hasil jaringan LSTM dengan *output* sebenarnya.
5. Perhitungan gradien untuk menentukan nilai *weight* dan *bias* supaya hasil *loss/error* mendekati 0 dengan menggunakan algoritma *backpropagation*.
6. Setelah mendapatkan nilai gradien, maka dilanjutkan dengan persamaan fungsi optimasi *Stochastic Gradient Descent* (SGD) dan update kembali nilai *weight* dan *bias*.
7. Jika seluruh *batch* data telah selesai di iterasi Kembali ke Langkah 2 hingga jumlah *epoch* yang telah ditentukan.

Berdasarkan Langkah-langkah *training* di atas, Pembentukan model LSTM diawali dengan model dibentuk dengan data dilatih dengan melewati mekanisme *Gates* pada LSTM. Data dilatih terus hingga mencapai batas *error* yang sesuai jumlah *epoch* yang diinginkan dengan penentuan serta peubahan *hyperparameter* yang digunakan.

#### **4.1.2.2. Proses *Testing Model***

Ketika pembelajaran sudah mencapai target sesuai nilai *epoch* yang di *inputkan*, proses iterasi berhenti dan berikutnya model diuji dengan data pengujian.

Pada proses ini memuatkan kembali *learned model* yang sudah dihasilkan pada proses *training* sebelumnya dan menghitung hasil keluaran berdasarkan nilai *bias* dan *weight* yang diberikan pada saat proses *training* yang sudah ada di dalam *learned model*. Kemudian dibandingkan dengan data yang dihasilkan dengan metode LSTM pada rentang waktu yang ditentukan dengan metode akurasi yang digunakan menggunakan RMSE.

#### **4.1.2.3. Perhitungan Manual LSTM**

Sebelum perancangan sistem dilakukan, peneliti melakukan analisis deskriptif untuk perancangan sistem dari Metode LSTM dengan perhitungan manual/numerik pada data Sebagian data yang di dapat langsung dari *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman*. Berikut contoh perhitungan manual jaringan LSTM.

##### **4.1.2.3.1. Inisialisasi *Hyperparameter***

Untuk melakukan perhitungan LSTM di perlukan menginisialisasi beberapa nilai *hyperparameter* yang perlu di tentukan sebelum tahap *training* data di lakukan seperti yang terlihat pada Tabel 4.6 berikut.

**Tabel 4.6. Nilai Inisialisasi Hyperparameter**

<i>Hyperparameter</i>	Nilai
Jumlah <i>Epoch</i>	50
Ukuran <i>Batch</i> ( $n$ )	1
Panjang <i>Timestep/Sequence</i>	2
<i>Features</i>	rr
Jumlah <i>Units LSTM</i>	1
Jumlah <i>Hidden Layer</i>	1
<i>Learning rate</i> ( $\gamma$ )	0.1
Probabilitas <i>Dropout</i>	0

Berdasarkan fungsi optimizer untuk perbaruan *weight* dan *bias* yang digunakan adalah *Stochastic Gradient Descent* (SGD), maka perbaruan *weight* dan *bias* akan dilakukan setiap proses *timestep* selesai, maka dari itu ukuran *batch* optimal adalah tidak lebih dari 1 seperti yang terlihat pada Tabel 4.6 di atas.

#### **4.1.2.3.2. Data Selection dan Preprocessing**

Untuk data yang digunakan sebanyak 60 baris data dengan *feature/variabel* data klimatologi yang digunakan dalam perhitungan manual ini terbatas hanya menggunakan *feature/variabel* (rr) curah hujan, dimulai dari rentang waktu 16 Oktober 2004 sampai 14 Desember 2004, *input* data *training* berbentuk tensor 3D di antaranya terdiri dari ukuran *batch* dimana hanya 1 *timestep*, jumlah *features* juga hanya 1 yaitu *feature/variabel* (rr) curah hujan, dan panjang *timestep/sequence* dengan panjang 2 baris data menjadi input  $\mathbf{x}_1$  dan  $\mathbf{x}_2$ , seperti yang terlihat pada Tabel 4.7 berikut.

**Tabel 4.7. Data Yang di Gunakan Dalam Perhitungan Manual LSTM**

No	Tanggal	rr	(rr) MinMaxScale	Input		y (label)
				x <sub>1</sub>	x <sub>2</sub>	
1	16 Oktober 2004	0.5	0.0052	0.0052	0.0206	1.0000
2	17 Oktober 2004	2	0.0206	0.0206	1.0000	0.5258
3	18 Oktober 2004	97	1.0000	1.0000	0.5258	0.6206
4	19 Oktober 2004	51	0.5258	0.5258	0.6206	0.2371
5	20 Oktober 2004	60.2	0.6206	0.6206	0.2371	0.1031
6	21 Oktober 2004	23	0.2371	0.2371	0.1031	0.1753
7	22 Oktober 2004	10	0.1031	0.1031	0.1753	0.0206
8	23 Oktober 2004	17	0.1753	0.1753	0.0206	0.0010
9	24 Oktober 2004	2	0.0206	0.0206	0.0010	0.1546
10	25 Oktober 2004	0.1	0.0010	0.0010	0.1546	0.0897
11	26 Oktober 2004	15	0.1546	0.1546	0.0897	0.2773
12	27 Oktober 2004	8.7	0.0897	0.0897	0.2773	0.1526
13	28 Oktober 2004	26.9	0.2773	0.2773	0.1526	0.4402
14	29 Oktober 2004	14.8	0.1526	0.1526	0.4402	0.1392
15	30 Oktober 2004	42.7	0.4402	0.4402	0.1392	0.0670
16	31 Oktober 2004	13.5	0.1392	0.1392	0.0670	0.1753
17	01 November 2004	6.5	0.0670	0.0670	0.1753	0.1897
18	02 November 2004	17	0.1753	0.1753	0.1897	0.2598
19	03 November 2004	18.4	0.1897	0.1897	0.2598	0.2959
20	04 November 2004	25.2	0.2598	0.2598	0.2959	0.2577
21	05 November 2004	28.7	0.2959	0.2959	0.2577	0.0103
22	06 November 2004	25	0.2577	0.2577	0.0103	0.1619
23	07 November 2004	1	0.0103	0.0103	0.1619	0.1959
24	08 November 2004	15.7	0.1619	0.1619	0.1959	0.0412
25	09 November 2004	19	0.1959	0.1959	0.0412	0.7021
26	10 November 2004	4	0.0412	0.0412	0.7021	0.0639
27	11 November 2004	68.1	0.7021	0.7021	0.0639	0.3093
28	12 November 2004	6.2	0.0639	0.0639	0.3093	0.6495
29	13 November 2004	30	0.3093	0.3093	0.6495	0.2268
30	14 November 2004	63	0.6495	0.6495	0.2268	0.5495
31	15 November 2004	22	0.2268	0.2268	0.5495	0.0320
32	16 November 2004	53.3	0.5495	0.5495	0.0320	0.0062
33	17 November 2004	3.1	0.0320	0.0320	0.0062	0.0031
34	18 November 2004	0.6	0.0062	0.0062	0.0031	0.0258
35	19 November 2004	0.3	0.0031	0.0031	0.0258	0.1299
36	20 November 2004	2.5	0.0258	0.0258	0.1299	0.8557
37	21 November 2004	12.6	0.1299	0.1299	0.8557	0.0103
38	22 November 2004	83	0.8557	0.8557	0.0103	0.3093
39	23 November 2004	1	0.0103	0.0103	0.3093	0.0144

40	24 November 2004	30	0.3093	0.3093	0.0144	0.0010
41	25 November 2004	1.4	0.0144	0.0144	0.0010	0.0474
42	26 November 2004	0.1	0.0010	0.0010	0.0474	0.5320
43	27 November 2004	4.6	0.0474	0.0474	0.5320	0.0515
44	28 November 2004	51.6	0.5320	0.5320	0.0515	0.0021
45	29 November 2004	5	0.0515	0.0515	0.0021	0.0938
46	30 November 2004	0.2	0.0021	0.0021	0.0938	0.7938
47	01 Desember 2004	9.1	0.0938	0.0938	0.7938	0.0165
48	02 Desember 2004	77	0.7938	0.7938	0.0165	0.1278
49	03 Desember 2004	1.6	0.0165	0.0165	0.1278	0.0000
50	04 Desember 2004	12.4	0.1278	0.1278	0.0000	0.4753
51	05 Desember 2004	0	0.0000	0.0000	0.4753	0.0206
52	06 Desember 2004	46.1	0.4753	0.4753	0.0206	0.0000
53	07 Desember 2004	2	0.0206			
54	08 Desember 2004	0	0.0000			
55	09 Desember 2004	14.5	0.1495	0.1495	0.0072	0.0082
56	10 Desember 2004	0.7	0.0072	0.0072	0.0082	0.0000
57	11 Desember 2004	0.8	0.0082	0.0082	0.0000	0.0175
58	12 Desember 2004	0	0.0000	0.0000	0.0175	0.0124
59	13 Desember 2004	1.7	0.0175			
60	14 Desember 2004	1.2	0.0124			

Seperti yang terlihat pada Tabel 4.7 dari 60 baris data yang di gunakan, data di bagi menjadi skala 9:1 untuk data *training* dan data *testing*, dimana baris data 1-54 di gunakan untuk data *training* dan baris data 55-60 di gunakan untuk data *testing*. Nilai dari *input* (x) dan label (y) di sesuaikan berdasarkan bentuk model yang di rancang, dikarenakan model menerima *input* berbentuk *timeseries* maka nilai dari tiap-tiap *input* dan label di dasarkan dengan bentuk pola data *sliding windows*, untuk ukuran tiap-tiap *input* di dasarkan pada ukuran *timestep*.

#### 4.1.2.3.3. Inisialisasi Dimensi Data dan Parameter

Inisialisasi Dimensi pada setiap *parameter* sangat di perlukan untuk pembangunan blok dasar dari jaringan LSTM termasuk jenis jaringan syaraf tiruan yang lainnya. Dimensi dari semua *parameter* dari LSTM tergantung pada dimensi *unit*

tersesembuni. Untuk penjelasan lebih lengkapnya dapat di lihat pada Tabel 4.8 berikut.

**Tabel 4.8. Dimensi Parameter Pada Jaringan LSTM**

Parameter	Keterangan	Dimensi
<i>Units</i>	Jumlah <i>neuron unit</i> dalam <i>hidden layer</i>	( <i>n_h</i> )
<i>feature</i>	Jumlah <i>Feature</i>	( <i>f</i> )
<i>n_x</i>	Ukuran <i>input</i> (panjang <i>timestep</i> )	( <i>n_x</i> )
$C^{<t-1>}$	Dimensi <i>cell state</i> sebelumnya	( <i>n_h, f</i> )
$h^{<t-1>}$	Dimensi <i>output</i> sebelumnya	( <i>n_h, f</i> )
$x^{<t>}$	<i>Input</i> saat ini	( <i>n_x, f</i> )
$[x^{<t>}, h^{<t-1>}]$	Gabungan <i>output</i> sebelumnya dan <i>input</i> saat ini	( <i>n_x + n_h, f</i> )
$W_f, W_i, W_c, W_o$	<i>Weight</i> untuk semua <i>gate</i>	( <i>n_h, n_x + n_h</i> )
$b_f, b_i, b_c, b_o$	<i>Bias</i> untuk semua operasi	( <i>n_h, 1</i> )

Seperti yang telihat pada Tabel 4.8 pada kolom Dimensi, Hampir setiap dimensi dari *parameter* di LSTM memiliki hubungan langsung maupun tidak langsung dengan *unit* tersembunyi (*n\_h*) pada *layer* LSTM, dan penting untuk di pahami bahwa perkalian matrik dari 2 metrik berukuran  $(a, b) * (b, c)$  maka menghasilkan *output* berukuran  $(a, c)$ . Setelah menganalisa dimensi komponen, tahap selanjutnya adalah menganalisa dimensi *output* pada setiap komponen. Untuk penjelasan lebih lengkapnya dapat di lihat pada Tabel 4.9 berikut.

**Tabel 4.9. Dimensi Output Pada Jaringan LSTM**

<b>Output</b>	<b>Operasi</b>	<b>Dimensi</b>
$f_t$	$\sigma(W_f * [x^{<t>} \ h^{<t-1>}] + b_f)$	$(n_h, n_x + n_h) * (n_x + n_h, m) + (n_h, 1) = (n_h, m)$
$i_t$	$\sigma(W_i * [x^{<t>} \ h^{<t-1>}] + b_i)$	$(n_h, n_x + n_h) * (n_x + n_h, m) + (n_h, 1) = (n_h, m)$
$\tilde{C}_t$	$\tanh(W_c * [x^{<t>} \ h^{<t-1>}] + b_c)$	$(n_h, n_x + n_h) * (n_x + n_h, m) + (n_h, 1) = (n_h, m)$
$o_t$	$\sigma(W_o * [x^{<t>} \ h^{<t-1>}] + b_o)$	$(n_h, n_x + n_h) * (n_x + n_h, m) + (n_h, 1) = (n_h, m)$
$C_t$	$f_t * C_{t-1} + i_t * \tilde{C}_t$	$(n_h, m) * (n_h, m) + (n_h, m) * (n_h, m) = (n_h, m)$
$h_t$	$o_t * \tanh(C_t)$	$(n_h, m) * (n_h, m) = (n_h, m)$

Berdasarkan Tabel 4.9 fungsi aktivasi *sigmoid* dan *tanh* di tetapkan sebagai elemen dari matrix sehingga dimensi *input* dan *output* tidak berubah. Setelah melakukan *preprocessing* data dari normalisasi, membagi data sampai membentuk data dengan pola *sliding windows* hingga menentukan nilai *hyperparameter* untuk model LSTM yang di gunakan. Tahap selanjutnya adalah menginisialisasi nilai awal dari *parameter forward step* seperti *bias* dan *weight*, *cell state* sebelumnya, dan *output* sebelumnya.

- Dimensi dari *Bias* seperti yang terlihat pada Tabel 4.8 berdimensi  $(n_h, 1)$ , di karenakan jumlah *unit* yang di gunakan adalah 1 maka bentuk dimensi menjadi  $(1, 1)$ , maka nilai *Bias* dari *forget*, *input*, *cell state*, *output* bisa di inisialisasikan dengan nilai nol seperti : [0], [0], [0], [0]

- Dimensi dari  $C_o$  seperti yang terlihat pada Tabel 4.8 berdimensi  $(n_h, f)$ , Dengan jumlah *unit* yang di gunakan adalah 1 dan jumlah *feature* adalah 1 maka bentuk dimensi menjadi  $(1, 1)$ , Untuk nilainya dikarenakan *Cell State* sebelumnya tidak ada maka nilai  $C_o$  di inisialisasi menjadi : [0]
- Dimensi dari  $h_0$  seperti yang terlihat pada Tabel 4.8 berdimensi  $(n_h, f)$ , Dengan jumlah *unit* yang di gunakan adalah 1 dan jumlah feature adalah 1 maka bentuk dimensi menjadi  $(1, 1)$ , Untuk nilainya dikarenakan *Output* sebelumnya tidak ada maka nilai  $h_0$  di inisialisasi menjadi : [0]
- Untuk nilai *Weight* seperti yang terlihat pada Tabel 4.8 berdimensi  $(n_h, n_x + n_h)$ , Dengan jumlah *unit* yang di gunakan adalah 1, Panjang *timestep* adalah 2 maka bentuk dimensi menjadi  $(1, 2+1)$ , maka nilai *Weight* untuk masing-masing *Weight forget, input, cell state, output* bisa di inisialisasikan dengan nilai acak atau rumus seperti berikut :

$$W = \left[ \pm \frac{1}{\sqrt{(n_x + n_h)}}, \pm \frac{1}{\sqrt{(n_x + n_h)}}, \pm \frac{1}{\sqrt{(n_x + n_h)}} \right]$$

$$W_f = [0.5774 \ 0.5774 \ 0.5774]$$

$$W_i = [0.5774 \ 0.5774 \ 0.5774]$$

$$W_c = [0.5774 \ 0.5774 \ 0.5774]$$

$$W_o = [0.5774 \ 0.5774 \ 0.5774]$$

Selanjutnya adalah menginisialisasi nilai awal dari *parameter backward* step seperti  $\Delta h_{n+1}$ ,  $f_{n+1}$ , dan  $\delta C_{n+1}$ .

- Untuk  $\Delta h_{n+1}$  di inisialisasi menjadi 0 di karenakan tidak ada *timestep* selanjutnya.
- Untuk  $f_{n+1}$  sama seperti sebelumnya untuk nilai dari  $f_{n+1}$  di inisialisasi menjadi 0.
- Untuk  $\delta C_{n+1}$  sama seperti sebelumnya untuk nilai dari  $\delta C_{n+1}$  di inisialisasi menjadi 0.

#### **4.1.2.3.4. Training Model LSTM**

Setelah di ketahui nilai *weight* awal kemudian barulah di mulai proses *training*.

Pada proses *training* algoritma pada model LSTM yang di gunakan *forward step* dan *backward step*. Berikut contoh perhitungan manual jaringan LSTM.

##### *4.1.2.3.4.1. Perhitungan Epoch 1 Batch 1*

Perhitungan di mulai dari *epoch* 1 dan *batch* 1 untuk satu kali iterasi yang di hitung secara numerik atau manual sebagai berikut.

###### *4.1.2.3.4.1.1. Forward Timestep ke 1 Batch 1*

Setelah menentukan *hyperparameter*, *parameter/nilai awal*, dan *dimensi* selanjutnya adalah menghitung forward step *timestep* 1 seperti berikut.

$$\begin{aligned}
 f_t &= \sigma(W_f * [x^{<t>} \ h^{<t-1>}]) + b_f \\
 &= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\
 &= \sigma([0.5774 \ 0.5774 \ 0.5774] * [0.0052 \ 0.0206 \ 0] + [0]) \\
 &= \sigma(0.0149) \\
 &= \frac{1}{1 + e^{-0.0149}} \\
 &= 0.5037
 \end{aligned}$$

$$\begin{aligned}
i_t &= \sigma(W_i * [x^{<t>} \ h^{<t-1>}]) + b_i \\
&= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\
&= \sigma([0.5774 \ 0.5774 \ -0.5774] * [0.0052 \ 0.0206 \ 0] + [0]) \\
&= \sigma(0.0149) \\
&= \frac{1}{1 + e^{-0.0149}} \\
&= 0.5037
\end{aligned}$$

$$\begin{aligned}
\tilde{C}_t &= \tanh(W_c * [x^{<t>} \ h^{<t-1>}]) + b_c \\
&= \tanh((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\
&= \tanh([0.5774 \ 0.5774 \ 0.5774] * [0.0052 \ 0.0206 \ 0] + [0]) \\
&= \tanh(0.0149) \\
&= 2 * \left( \frac{1}{1 + e^{-2*0.0149}} \right) - 1 \\
&= 0.0149
\end{aligned}$$

$$\begin{aligned}
o_t &= \sigma(W_o * [x^{<t>} \ h^{<t-1>}]) + b_o \\
&= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\
&= \sigma([0.5774 \ 0.5774 \ 0.5774] * [0.0052 \ 0.0206 \ 0] + [0]) \\
&= \sigma(0.0149) \\
&= \frac{1}{1 + e^{-0.0149}} \\
&= 0.5037
\end{aligned}$$

$$\begin{aligned}
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
&= 0.5037 * 0 + 0.5037 * 0.0149 \\
&= 0.0075
\end{aligned}$$

$$h_t = o_t * \tanh(C_t)$$

$$\begin{aligned}
&= 0.5037 * \tanh(0.0075) \\
&= 0.5037 * \left( 2 * \left( \frac{1}{1 + e^{-2*0.0075}} \right) - 1 \right) \\
&= 0.0037
\end{aligned}$$

#### 4.1.2.3.4.1.2. Backward Timestep ke 1 Batch I

Dikarenakan ukuran batch tidak lebih dari 1 maka setiap forward step selesai dalam 1 *timestep* langsung dilakukan backward step.

$$\begin{aligned}
\delta h_t &= h_t - y^{<t>} + \Delta h_{t+1} \\
&= 0.0037 - 1.0 + 0 \\
&= -0.9962 \\
\delta C_t &= \delta h_t * o_t * (1 - \tanh(C_t)^2) + \delta C_{t+1} * f_{t+1} \\
&= -0.9962 * 0.5037 * (1 - \tanh(0.0075)^2) + 0 * 0 \\
&= -0.5018 \\
\delta o_t &= \delta h_t * \tanh(C_t) * o_t * (1 - o_t) \\
&= -0.9962 * \tanh(0.0075) * 0.5037 * (1 - 0.5037) \\
&= -0.0019 \\
\widetilde{\delta C}_t &= \delta C_t * i_t * \left( 1 - \tilde{C}_t^2 \right) \\
&= -0.5018 * 0.5037 * \left( 1 - 0.0149^2 \right) \\
&= -0.2527 \\
\delta i_t &= \delta C_t * \tilde{C}_t * i_t * (1 - i_t) \\
&= -0.5018 * 0.0149 * 0.5037 * (1 - 0.5037) \\
&= -0.0019 \\
\delta f_t &= \delta C_t * C_{t-1} * f_t * (1 - f_t)
\end{aligned}$$

$$= -0.5018 * 0 * 0.5037 * (1 - 0.5037)$$

$$= 0.0000$$

$$U_f = \text{weight } h \text{ dari } W_f [x_1 \ x_2 \ h] [0.5774 \ 0.5774 \ 0.5774]$$

$$= 0.5774$$

$$U_i = \text{weight } h \text{ dari } W_i [x_1 \ x_2 \ h] [0.5774 \ 0.5774 \ 0.5774]$$

$$= 0.5774$$

$$U_c = \text{weight } h \text{ dari } W_c [x_1 \ x_2 \ h] [0.5774 \ 0.5774 \ 0.5774]$$

$$= 0.5774$$

$$U_o = \text{weight } h \text{ dari } W_o [x_1 \ x_2 \ h] [0.5774 \ 0.5774 \ 0.5774]$$

$$= 0.5774$$

$$\Delta h_t = [U_f \ U_i \ U_c \ U_o] * [\delta f_t \ \delta i_t \ \delta \tilde{C}_t \ \delta o_t]$$

$$= \begin{bmatrix} 0.5774 \\ 0.5774 \\ 0.5774 \\ 0.5774 \end{bmatrix} * \begin{bmatrix} 0.0000 \\ -0.0019 \\ -0.2527 \\ -0.0019 \end{bmatrix}$$

$$= -0.1481$$

#### 4.1.2.3.4.1.3. Perbaharui Bias dan Weight Batch 1

Berdasarkan optimizer yang di gunakan adalah *Stochastic Gradient Descent* (SDG), maka perbaharuan *bias* dan *weight* di lakukan di setiap *timestep*.

$$\begin{aligned} W_f^{new} &= W_f^{old} - \gamma * \sum_{t=1}^n [\delta f_t] * [x^{<t>} \ h^{<t-1>}] \\ &= W_f^{old} - 0.1 * ([0.0000] * [0.0052 \ 0.0206 \ 0.0000]) \\ &= [0.5774 \ 0.5774 \ 0.5774] - 0.1 * [0.0000 \ 0.0000 \ 0.0000] \\ &= [0.5774 \ 0.5774 \ 0.5774] \end{aligned}$$

$$\begin{aligned}
b_f^{new} &= b_f^{old} - \gamma * \sum_{t=1}^n [\delta f_t] \\
&= 0 - 0.1 * ([0.0000]) \\
&= 0.0000 \\
W_i^{new} &= W_i^{old} - \gamma * \sum_{t=1}^n [\delta i_t] * [x^{<t>} \ h^{<t-1>}] \\
&= W_i^{old} - 0.1 * ([-0.0019] * [0.0052 \ 0.0206 \ 0.0000]) \\
&= [0.5774 \ 0.5774 \ 0.5774] - 0.1 * [-0.00001 \ -0.00004 \ 0.000] \\
&= [0.577401 \ 0.577404 \ 0.5774] \\
b_i^{new} &= b_i^{old} - \gamma * \sum_{t=1}^n [\delta i_t] \\
&= 0 - 0.1 * ([-0.0019]) \\
&= 0.0002 \\
W_c^{new} &= W_c^{old} - \gamma * \sum_{t=1}^n [\delta C_t] * [x^{<t>} \ h^{<t-1>}] \\
&= W_c^{old} - 0.1 * ([-0.2527] * [0.0052 \ 0.0206 \ 0.0000]) \\
&= [0.5774 \ 0.5774 \ 0.5774] - 0.1 * [-0.0013 \ -0.0052 \ 0.0000] \\
&= [0.57753 \ 0.57792 \ 0.5774] \\
b_c^{new} &= b_c^{old} - \gamma * \sum_{t=1}^n [\delta C_t] \\
&= 0 - 0.1 * ([-0.2527]) \\
&= 0.0253 \\
W_o^{new} &= W_o^{old} - \gamma * \sum_{t=1}^n [\delta o_t] * [x^{<t>} \ h^{<t-1>}]
\end{aligned}$$

$$\begin{aligned}
&= W_o^{old} - 0.1 * ([-0.0019] * [0.0052 \quad 0.0206 \quad 0.0000]) \\
&= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 \\
&\quad * [-0.00001 \quad -0.00004 \quad 0.0000] \\
&= [0.577401 \quad 0.577404 \quad 0.5774] \\
b_o^{new} &= b_o^{old} - \gamma * \sum_{t=1}^n [\delta o_t] \\
&= 0 - 0.1 * ([-0.0019]) \\
&= 0.0002
\end{aligned}$$

#### 4.1.2.3.4.1.4. Hitung Akurasi Batch 1

Nilai Akurasi di hitung menggunakan rumus MSE, RMSE dari perhitungan kereluruhan timeteps pada batch 1 seperti berikut.

$$\begin{aligned}
MSE &= \frac{\sum_{t=1}^n (y_t - h_t)^2}{n} \\
&= \frac{(1.0000 - 0.0037)^2}{1} \\
&= 0.992463094
\end{aligned}$$

$$\begin{aligned}
RMSE &= \sqrt{MSE} \\
&= 0.996224419
\end{aligned}$$

$$\begin{aligned}
x_{denorm} &= x_{norm}(x_{max} - x_{min}) + x_{min} \\
&= 0.996224419 * (97 - 0) + 0 \\
&= 96.3
\end{aligned}$$

Berdasarkan hasil nilai error RMSE dari data *batch 1* yang di normalisasi dengan minmaxscale di dapat nilai error sebesar 0.996224419 maka di perlukan

untuk mengubah ke nilai real dengan cara denormalisasi nilai error dari hasil RMSE tersebut sehingga di dapat nilai error sebenarnya dengan nilai 96.3. Dimana nilai error 96.3 memiliki arti bahwa prediksi *feature* (*rr*) memiliki tingkat ketidakakuratan lebih kurang sebesar 96.3 dalam memprediksi curah hujan.

#### *4.1.2.3.4.2. Perhitungan Epoch 1 Batch 2-52*

Perhitungan *batch* 2-52 sama seperti sebelumnya, hanya saja beberapa *parameter* seperti *bias* dan *weight* di perbarui setelah menghitung *Stochastic Gradient Descent* (SGD) dari perhitungan *batch* sebelumnya.

##### *4.1.2.3.4.2.1. Forward Epoch 1 Batch 2-52*

*Forward batch* 2-52 berikut hasil perhitungan *forward* pada *batch* 2-52 seperti yang terlihat pada Tabel 4.10 berikut.

**Tabel 4.10. Hasil *Forward Epoch 1 Batch 2-52***

<i>batch</i>	<i>time step</i>	<i>Forward</i>					
		<i>Gate</i>				<i>c<sub>t</sub></i>	<i>h<sub>t</sub></i>
		<i>f<sub>t</sub></i>	<i>i<sub>t</sub></i>	<i>̃c<sub>t</sub></i>	<i>o<sub>t</sub></i>		
2	1	0.6432	0.6432	0.5477	0.6432	0.3523	0.2177
3	1	0.7070	0.7078	0.7254	0.7078	0.5134	0.3345
4	1	0.6597	0.6615	0.6108	0.6618	0.4041	0.2537
5	1	0.6213	0.6230	0.4924	0.6232	0.3067	0.1854
6	1	0.5490	0.5502	0.2301	0.5503	0.1266	0.0693
7	1	0.5401	0.5414	0.1992	0.5416	0.1079	0.0582
8	1	0.5282	0.5294	0.1508	0.5296	0.0798	0.0422
9	1	0.5031	0.5042	0.0492	0.5043	0.0248	0.0125
10	1	0.5225	0.5237	0.1310	0.5238	0.0686	0.0359
11	1	0.5352	0.5365	0.1824	0.5366	0.0978	0.0523
12	1	0.5528	0.5543	0.2575	0.5545	0.1427	0.0786
13	1	0.5617	0.5633	0.2928	0.5635	0.1649	0.0921
14	1	0.5847	0.5868	0.3878	0.5870	0.2276	0.1313
15	1	0.5829	0.5849	0.3800	0.5851	0.2223	0.1279
16	1	0.5297	0.5314	0.1774	0.5316	0.0943	0.0500
17	1	0.5349	0.5368	0.2017	0.5369	0.1083	0.0579

18	1	0.5525	0.5545	0.2731	0.5547	0.1514	0.0833
19	1	0.5645	0.5667	0.3234	0.5669	0.1833	0.1028
20	1	0.5795	0.5820	0.3833	0.5823	0.2231	0.1278
21	1	0.5792	0.5819	0.3856	0.5821	0.2244	0.1285
22	1	0.5386	0.5409	0.2293	0.5411	0.1240	0.0668
23	1	0.5248	0.5272	0.1788	0.5274	0.0943	0.0496
24	1	0.5515	0.5540	0.2859	0.5542	0.1584	0.0871
25	1	0.5342	0.5366	0.2173	0.5368	0.1166	0.0623
26	1	0.6057	0.6090	0.4934	0.6092	0.3005	0.1777
27	1	0.6088	0.6118	0.4994	0.6120	0.3055	0.1814
28	1	0.5537	0.5567	0.3097	0.5569	0.1724	0.0951
29	1	0.6350	0.6389	0.5953	0.6392	0.3804	0.2320
30	1	0.6239	0.6277	0.5616	0.6280	0.3525	0.2127
31	1	0.6102	0.6148	0.5277	0.6151	0.3244	0.1929
32	1	0.5832	0.5874	0.4346	0.5877	0.2553	0.1469
33	1	0.5055	0.5090	0.1361	0.5092	0.0693	0.0352
34	1	0.5013	0.5048	0.1184	0.5049	0.0598	0.0301
35	1	0.5042	0.5076	0.1298	0.5078	0.0659	0.0334
36	1	0.5225	0.5261	0.2057	0.5263	0.1082	0.0567
37	1	0.6385	0.6432	0.6213	0.6435	0.3997	0.2443
38	1	0.6225	0.6267	0.5700	0.6269	0.3572	0.2149
39	1	0.5460	0.5499	0.3132	0.5501	0.1722	0.0938
40	1	0.5466	0.5506	0.3140	0.5508	0.1729	0.0943
41	1	0.5022	0.5058	0.1376	0.5059	0.0696	0.0351
42	1	0.5070	0.5105	0.1570	0.5107	0.0802	0.0409
43	1	0.5829	0.5869	0.4517	0.5871	0.2651	0.1521
44	1	0.5834	0.5876	0.4520	0.5878	0.2656	0.1526
45	1	0.5077	0.5112	0.1658	0.5113	0.0848	0.0432
46	1	0.5138	0.5173	0.1913	0.5174	0.0990	0.0510
47	1	0.6254	0.6295	0.5942	0.6297	0.3741	0.2251
48	1	0.6149	0.6189	0.5597	0.6191	0.3464	0.2063
49	1	0.5208	0.5242	0.2305	0.5243	0.1208	0.0630
50	1	0.5184	0.5218	0.2197	0.5219	0.1147	0.0596
51	1	0.5682	0.5718	0.4131	0.5719	0.2362	0.1326
52	1	0.5711	0.5749	0.4217	0.5750	0.2424	0.1367

Pada Tabel 4.10 memperlihatkan hasil forward step LSTM pada epoch 1 dari setiap *timestep* di dalam batch 2-52 dengan ukuran batch tidak lebih dari 1 *timestep*.

#### 4.1.2.3.4.2.2. Backward Epoch 1 Batch 2-52

Backward batch 2-52 berikut hasil perhitungan *backward* pada batch 2-52 seperti yang terlihat pada Tabel 4.11 berikut.

**Tabel 4.11. Hasil Backward Epoch 1 Batch 2-52**

<i>batch</i>	time step	<i>Backward</i>							$\Delta h_t$	
		$\delta h_t$	$\delta C_t$	<i>Gate</i>						
				$\delta o_t$	$\widetilde{\delta C}_t$	$\delta i_t$	$\delta f_t$			
2	1	-0.3081	-0.1755	-0.0239	-0.0790	-0.0221	0.0000	-0.0722		
3	1	-0.2861	-0.1573	-0.0280	-0.0528	-0.0236	0.0000	-0.0602		
4	1	0.0166	0.0094	0.0014	0.0039	0.0013	0.0000	0.0038		
5	1	0.0823	0.0467	0.0057	0.0221	0.0054	0.0000	0.0192		
6	1	-0.1060	-0.0574	-0.0033	-0.0299	-0.0033	0.0000	-0.0211		
7	1	0.0376	0.0201	0.0010	0.0105	0.0010	0.0000	0.0072		
8	1	0.0411	0.0217	0.0008	0.0112	0.0008	0.0000	0.0074		
9	1	-0.1421	-0.0716	-0.0009	-0.0360	-0.0009	0.0000	-0.0218		
10	1	-0.0538	-0.0281	-0.0009	-0.0144	-0.0009	0.0000	-0.0094		
11	1	-0.2250	-0.1196	-0.0055	-0.0620	-0.0054	0.0000	-0.0421		
12	1	-0.0740	-0.0402	-0.0026	-0.0208	-0.0026	0.0000	-0.0150		
13	1	-0.3481	-0.1909	-0.0140	-0.0983	-0.0137	0.0000	-0.0728		
14	1	-0.0078	-0.0044	-0.0004	-0.0022	-0.0004	0.0000	-0.0017		
15	1	0.0609	0.0339	0.0032	0.0170	0.0031	0.0000	0.0135		
16	1	-0.1253	-0.0660	-0.0029	-0.0340	-0.0029	0.0000	-0.0230		
17	1	-0.1318	-0.0699	-0.0035	-0.0360	-0.0035	0.0000	-0.0249		
18	1	-0.1764	-0.0957	-0.0065	-0.0491	-0.0065	0.0000	-0.0359		
19	1	-0.1931	-0.1059	-0.0086	-0.0537	-0.0084	0.0000	-0.0408		
20	1	-0.1299	-0.0720	-0.0069	-0.0358	-0.0067	0.0000	-0.0285		
21	1	0.1182	0.0654	0.0063	0.0324	0.0061	0.0000	0.0259		
22	1	-0.0951	-0.0507	-0.0029	-0.0260	-0.0029	0.0000	-0.0183		
23	1	-0.1463	-0.0765	-0.0034	-0.0390	-0.0034	0.0000	-0.0265		
24	1	0.0458	0.0248	0.0018	0.0126	0.0017	0.0000	0.0093		
25	1	-0.6397	-0.3388	-0.0185	-0.1732	-0.0183	0.0000	-0.1212		
26	1	0.1138	0.0634	0.0079	0.0292	0.0075	0.0000	0.0257		
27	1	-0.1279	-0.0714	-0.0090	-0.0328	-0.0085	0.0000	-0.0290		
28	1	-0.5544	-0.2997	-0.0234	-0.1509	-0.0229	0.0000	-0.1138		
29	1	0.0052	0.0029	0.0004	0.0012	0.0004	0.0000	0.0012		
30	1	-0.3368	-0.1873	-0.0266	-0.0805	-0.0246	0.0000	-0.0760		
31	1	0.1609	0.0892	0.0119	0.0396	0.0112	0.0000	0.0362		

32	1	0.1407	0.0775	0.0085	0.0369	0.0082	0.0000	0.0310
33	1	0.0321	0.0163	0.0006	0.0081	0.0006	0.0000	0.0053
34	1	0.0044	0.0022	0.0001	0.0011	0.0001	0.0000	0.0007
35	1	-0.0965	-0.0488	-0.0016	-0.0243	-0.0016	0.0000	-0.0159
36	1	-0.7990	-0.4156	-0.0215	-0.2094	-0.0213	0.0000	-0.1456
37	1	0.2340	0.1289	0.0204	0.0509	0.0184	0.0000	0.0518
38	1	-0.0944	-0.0522	-0.0076	-0.0221	-0.0070	0.0000	-0.0212
39	1	0.0794	0.0424	0.0034	0.0210	0.0033	0.0000	0.0160
40	1	0.0933	0.0499	0.0040	0.0247	0.0039	0.0000	0.0188
41	1	-0.0123	-0.0062	-0.0002	-0.0031	-0.0002	0.0000	-0.0020
42	1	-0.4911	-0.2492	-0.0098	-0.1241	-0.0098	0.0000	-0.0830
43	1	0.1006	0.0551	0.0063	0.0257	0.0060	0.0000	0.0220
44	1	0.1505	0.0825	0.0095	0.0386	0.0090	0.0000	0.0330
45	1	-0.0506	-0.0257	-0.0011	-0.0128	-0.0011	0.0000	-0.0086
46	1	-0.7428	-0.3806	-0.0183	-0.1897	-0.0182	0.0000	-0.1306
47	1	0.2087	0.1146	0.0174	0.0467	0.0159	0.0000	0.0462
48	1	0.0784	0.0432	0.0062	0.0184	0.0057	0.0000	0.0174
49	1	0.0630	0.0326	0.0019	0.0162	0.0019	0.0000	0.0115
50	1	-0.4157	-0.2141	-0.0118	-0.1063	-0.0117	0.0000	-0.0750
51	1	0.1120	0.0606	0.0064	0.0287	0.0061	0.0000	0.0238
52	1	0.1367	0.0742	0.0079	0.0351	0.0076	0.0000	0.0292

Pada Tabel 4.11 memperlihatkan hasil backward step LSTM pada epoch 1 dari setiap *timestep* di dalam batch 2-52 dengan ukuran batch tidak lebih dari 1 *timestep*.

#### 4.1.2.3.4.2.3. Hitung Error Epoch 1 Batch 2-52

Nilai Error di hitung menggunakan rumus MSE, RMSE dari perhitungan kereluruhan timeteps pada *batch* 2-52 seperti pada Tabel 4.12 berikut.

**Tabel 4.12. Nilai Error Epoch 1 Batch 2-52**

<i>Batch</i>	MSE	RMSE	RMSE (Denormalisasi)
2	0.094914	0.308082	29.9
3	0.081859	0.286109	27.8
4	0.000277	0.016633	1.6

5	0.006772	0.082291	8.0
6	0.011227	0.105957	10.3
7	0.001411	0.037566	3.6
8	0.001693	0.041148	4.0
9	0.020198	0.142121	13.8
10	0.002895	0.053804	5.2
11	0.050617	0.224982	21.8
12	0.005471	0.073968	7.2
13	0.121185	0.348116	33.8
14	0.000061	0.007836	0.8
15	0.003713	0.060935	5.9
16	0.015694	0.125276	12.2
17	0.017367	0.131783	12.8
18	0.031134	0.176448	17.1
19	0.037297	0.193125	18.7
20	0.016886	0.129948	12.6
21	0.013961	0.118156	11.5
22	0.009040	0.095078	9.2
23	0.021407	0.146310	14.2
24	0.002100	0.045822	4.4
25	0.409278	0.639748	62.1
26	0.012956	0.113823	11.0
27	0.016365	0.127924	12.4
28	0.307377	0.554416	53.8
29	0.000028	0.005245	0.5
30	0.113453	0.336828	32.7
31	0.025888	0.160898	15.6
32	0.019787	0.140666	13.6
33	0.001032	0.032129	3.1
34	0.000019	0.004370	0.4
35	0.009309	0.096485	9.4
36	0.638327	0.798954	77.5
37	0.054753	0.233993	22.7
38	0.008914	0.094413	9.2
39	0.006300	0.079373	7.7
40	0.008697	0.093258	9.0
41	0.000151	0.012275	1.2
42	0.241178	0.491098	47.6
43	0.010111	0.100555	9.8
44	0.022649	0.150496	14.6
45	0.002558	0.050572	4.9

46	0.551707	0.742770	72.0
47	0.043536	0.208653	20.2
48	0.006154	0.078447	7.6
49	0.003972	0.063027	6.1
50	0.172783	0.415672	40.3
51	0.012546	0.112008	10.9
52	0.018693	0.136723	13.3

Berdasarkan Tabel 4.12 hasil nilai error RMSE dari epoch 1 yang di normalisasi dengan minmaxscale di dapat nilai error seperti yang terlihat pada kolom RMSE (Denormalisasi) memiliki arti bahwa prediksi feature (rr) setiap batch memiliki tingkat ketidakakuratan lebih kurang seperti yang terlihat pada kolom tersebut dalam memprediksi curah hujan.

#### 4.1.2.3.4.2.4. Hitung Total Nilai Error Epoch 1

Untuk mengetahui total error dari 1 kali *epoch* diperlukan untuk mencari nilai rata-rata dari total semua nilai error dari tiap *batch*.

$$\sum MSE = \frac{0.996224 + 0.308082 + 0.286109 + 0.016633 + 0.082291 + 0.105957 + 0.037566 + 0.041148 + 0.142121 + 0.053804 + 0.224982 + 0.073968 + 0.348116 + 0.007836 + 0.060935 + 0.125276 + 0.131783 + 0.176448 + 0.193125 + 0.129948 + 0.118156 + 0.095078 + 0.14631 + 0.045822 + 0.639748 + 0.113823 + 0.127924 + 0.554416 + 0.005245 + 0.336828 + 0.160898 + 0.140666 + 0.032129 + 0.00437 + 0.096485 + 0.798954 + 0.233993 + 0.094413 + 0.079373 + 0.093258 + 0.012275 + 0.491098 + 0.100555 + 0.150496 + 0.050572 + 0.74277 + 0.208653 + 0.078447 + 0.063027 + 0.415672 + 0.112008 + 0.136723}{52}$$

$$= 0.082272329$$

$$\sum RMSE = \sqrt{\sum MSE}$$

$$= 0.286831535$$

$$\begin{aligned}
 x_{denorm} &= x_{norm}(x_{max} - x_{min}) + x_{min} \\
 &= 0.286831535 * (97 - 0) + 0 \\
 &= 27.8
 \end{aligned}$$

Berdasarkan hasil total nilai error RMSE dari data *epoch* 1 yang di normalisasi dengan minmaxscale di dapat nilai error sebesar 0.286831535 maka di perlukan untuk mengubah ke nilai real dengan cara denormalisasi nilai error dari hasil RMSE tersebut sehingga di dapat nilai error sebenarnya dengan nilai 27.8. Dimana nilai error 27.8 memiliki arti bahwa prediksi *feature* (*rr*) memiliki tingkat ketidakakuratan lebih kurang sebesar 27.8 dalam memprediksi curah hujan.

#### 4.1.2.3.4.3. Perhitungan Epoch 2-50

Pada perhitungan *Epoch* 2-50 nilai error di hitung menggunakan total dari rumus MSE, RMSE seperti pada Tabel 4.13 berikut.

**Tabel 4.13. Nilai Error Epoch 2-50**

<i>Epoch</i>	$\sum MSE$	$\sum RMSE$	$\Sigma RMSE$ (Denormalisasi)
2	0.075853	0.275413	26.7
3	0.072705	0.269638	26.2
4	0.071165	0.266767	25.9
5	0.070256	0.265059	25.7
6	0.069687	0.263983	25.6
7	0.069310	0.263267	25.5
8	0.069046	0.262766	25.5
9	0.068851	0.262395	25.5
10	0.068700	0.262107	25.4
11	0.068577	0.261872	25.4
12	0.068472	0.261672	25.4
13	0.068379	0.261494	25.4
14	0.068295	0.261333	25.3
15	0.068218	0.261185	25.3
16	0.068144	0.261043	25.3

17	0.068073	0.260907	25.3
18	0.068004	0.260776	25.3
19	0.067938	0.260649	25.3
20	0.067873	0.260525	25.3
21	0.067696	0.260185	25.2
22	0.067525	0.259856	25.2
23	0.067362	0.259542	25.2
24	0.067206	0.259241	25.1
25	0.067057	0.258953	25.1
26	0.066913	0.258676	25.1
27	0.066776	0.258410	25.1
28	0.066643	0.258154	25.0
29	0.066516	0.257907	25.0
30	0.066394	0.257670	25.0
31	0.066294	0.257476	25.0
32	0.066197	0.257288	25.0
33	0.066104	0.257106	24.9
34	0.066013	0.256930	24.9
35	0.065926	0.256760	24.9
36	0.065841	0.256594	24.9
37	0.065758	0.256433	24.9
38	0.065678	0.256277	24.9
39	0.065600	0.256126	24.8
40	0.065525	0.255978	24.8
41	0.065461	0.255853	24.8
42	0.065397	0.255728	24.8
43	0.065334	0.255606	24.8
44	0.065274	0.255488	24.8
45	0.065215	0.255372	24.8
46	0.065157	0.255259	24.8
47	0.065101	0.255149	24.7
48	0.065046	0.255042	24.7
49	0.064993	0.254936	24.7
50	0.064940	0.254834	24.7

Berdasarkan Tabel 4.13 hasil nilai error RMSE dari epoch 2-52 yang di normalisasi dengan minmaxscale di dapat nilai error seperti yang terlihat pada kolom  $\Sigma RMSE$  (Denormalisasi), dimana nilai error selalu berkurang seiring

dengan perulangan epoch selama 50 kali. memiliki arti bahwa prediksi feature (rr) setiap perulangan epoch yang di lakukan memiliki tingkat ketidakakuratan terus berkurang seperti yang terlihat pada kolom tersebut dalam memprediksi curah hujan.

#### **4.1.2.3.5. Testing Model LSTM**

Setelah melakukan *Training* Model LSTM maka di ketahui nilai *bias* dan *weight* optimal dari 50 *epoch* seperti pada Tabel 4.14 berikut.

**Tabel 4.14. Bias dan Weight dari Learned Model**

Parameters	Forget Gate	Input Gate	Memory Gate	Output Gate
<i>bias</i>	[0.0000]	[-0.0569]	[0.9422]	[-0.1381]
<i>weight</i>	[0.5774	[0.5023	[0.5979	[0.4963
	0.5774	0.4082	0.5563	0.3731
	0.5774]	0.5774]	0.5774	0.5574]

Hasil optimal dari *bias* dan *weight* dari learned model di gunakan untuk melakukan pengujian dari data *testing*. Pada proses *testing* algoritma pada model LSTM yang di gunakan hanya *forward* step sebanyak data yang di *testing* dalam 1 kali iterasi seperti yang terlihat pada Tabel 4.15.

**Tabel 4.15. Hasil Testing Model**

<i>t</i>	<i>y<sub>t</sub></i>	<i>y<sub>t</sub></i> (Denormalisasi)	<i>h<sub>t</sub></i>	<i>h<sub>t</sub></i> (Denormalisasi)
1	0.0082	0.8	0.1809	17.5
2	0.0000	0	0.1617	15.7
3	0.0175	1.7	0.1609	15.6
4	0.0124	1.2	0.1617	15.7

Pada Tabel 4.15 untuk nilai error sama seperti pada tahap *training*, perbedaan nya untuk data denormalisasi peneliti hanya menggunakan rumus RMSE untuk perhitungan nilai error.

$$\begin{aligned}
 RMSE &= \sqrt{\frac{\sum_{t=1}^n (y_t - h_t)^2}{n}} \\
 &= \sqrt{\frac{(0.8 - 17.5)^2 + (0 - 15.7)^2 + (1.7 - 15.6)^2 + (1.2 - 15.7)^2}{4}} \\
 &= 15.24
 \end{aligned}$$

Berdasarkan hasil nilai error RMSE dari proses *testing* di dapat nilai error sebesar 15.24. Dimana nilai error 15.24 memiliki arti bahwa prediksi *feature (rr)* pada saat *testing* memiliki tingkat ketidakakuratan lebih kurang sebesar 15.24 dalam memprediksi *feature/variable (rr)* curah hujan. Dapat dikatakan bahwa Proyeksi atau perhitungan manual LSTM yang dilakukan peneliti dengan hanya menggunakan 1 *feature/variabel* yaitu (*rr*) curah hujan, dengan beberapa nilai *hyperparameter* lainnya seperti yang terlihat pada Tabel 4.6, dan menggunakan *deep learning* dengan metode *Long Short-Term Memory* dengan algoritma *backpropagation* masih jauh dari nilai keakuratan yang di harapkan.

#### **4.1.3. Analisa Sistem**

Analisa ini dilakukan untuk mengetahui apa saja yang dibutuhkan dalam perancangan sistem dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, dan hambatan yang muncul. Sehingga menghasilkan sebuah sistem yang efektif dan efisien. Sistem yang dikembangkan berbasis web dengan Bahasa Pemrograman Python dan *database MySQL*.

Sistem yang dikembangkan Bernama “Sistem Proyeksi Curah Hujan Padang Pariaman”. Sistem ini dirancang untuk melakukan pembelajaran mendalam

dengan mengalkulasi dari kumpulan data klimatologi hingga menghasilkan sebuah prediksi sesuai yang di harapkan.

Data yang di dapat dari *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* masih di dokumentasi dalam bentuk *format excel* yang nantinya semua data tersebut di masukkan ke dalam *database MySQL* yang di gunakan untuk sistem dalam melakukan pembelajaran mendalam.

Keuntungan dari Sistem ini adalah menghilangkan kebiasaan melakukan pemodelan prediksi dengan penggunaan jumlah *parameter*, asumsi-asumsi matematis, dan *formulasi* persamaan yang cenderung rumit. Karena untuk menghasilkan model dengan prediksi yang tepat di butuhkah banyak *parameter* yang mustahil di lakukan secara manual.

#### **4.1.3.1. Analisa Desain Database**

*Database* adalah kumpulan dari beberapa tabel dependen maupun independen satu sama lain dengan tabel yang lainnya. Tabel-tabel yang dependen dengan tabel lainya terhubung berdasarkan primary key yang ada. Seperti yang di jabarkan sebagai berikut :

##### **4.1.3.1.1. Tabel Users**

Merupakan tabel yang di gunakan untuk menampung data penggunaan untuk proses authentikasi masuk ke sistem dengan rancangan struktur tabel seperti pada Tabel 4.16 berikut:

- Nama *Database* : proyeksi
- Nama Tabel : *auth\_user*

**Tabel 4.16. Tabel *auth\_user***

No	Field Name	Type	Width	Description
1	id	Integer	11	Primary Key
2	password	Varchar	128	Password <i>User</i>
3	last_login	Datetime	-	<i>Login</i> Terakhir <i>User</i>
4	is_superuser	Bool	2	Apakah Super <i>User</i>
5	username	Varchar	150	Panggilan <i>User</i>
6	last_name	Varchar	150	Nama Terakhir <i>User</i>
7	email	Varchar	254	Email <i>User</i>
8	is_staff	Bool	2	Apakah Staff
9	is_active	Bool	2	Apakah Aktif
10	date_joined	Datetime	-	Waktu Daftar ke Sistem
11	first_name	Varchar	150	Nama Depan <i>User</i>

Pada Tabel 4.16 telihat seperti pola table authentikasi yang kerap digunakan pada sistem pada umumnya, dimana tabel tersebut memiliki, password, *username*, email,dan date\_joined, sebagai inti dari kolom informasi authentikasi, dan beberapa kolom lainnya sebagai pendukung/fitur tambahan.

#### **4.1.3.1.2. Tabel Session Middleware**

Merupakan tabel yang di gunakan untuk menampung data sesi penggunaan yang telah berhasil melakukan authentikasi ke dalam sistem dengan rancangan struktur tabel seperti pada Tabel 4.17 berikut:

- Nama *Database* : proyeksi
- Nama Tabel : django\_session

**Tabel 4.17. Tabel *django\_session***

No	Field Name	Type	Width	Description
1	session_key	Varchar	40	Primary Key
2	session_data	Text	-	Data Session <i>User</i>
3	expire_date	Datetime	-	Waktu Kadaluarsa Session <i>User</i>

Pada Tabel 4.17 memiliki beberapa kolom untuk menyimpan informasi ketika *admin* dalam sesi *login* ke dalam sistem, dimana tabel tersebut memiliki kolom session\_data untuk menyimpan data sesi *login admin*, dan expire\_date menyimpan informasi waktu kadaluarsa *admin* dalam sesi *login*.

#### **4.1.3.1.3. Tabel Klimatologi**

Merupakan tabel yang di gunakan untuk menampung data Klimatologi yang nantinya di gunakan dalam pembelajaran mesin dalam sistem dengan rancangan struktur tabel seperti pada Tabel 4.18 berikut :

- Nama *Database* : proyeksi
- Nama Tabel : proyeksi\_klimatologi

**Tabel 4.18. Tabel *proyeksi\_klimatologi***

No	Field Name	Type	Width	Description
1	id	Integer	11	Primary Key
2	tanggal	Date	-	Tanggal Observasi
3	tn	Double	4	Temperatur Minimum
4	tx	Double	4	Temperatur Maksimum
5	tavg	Double	4	Temperatur Rata-rata

6	rh_avg	Double	4	Kelembapan Rata-rata
7	rr	Double	4	Curah Hujan
8	ff_x	Double	4	Lama Penyinaran Matahari
9	ddd_x	Double	4	Kecepatan Angin Maksimum
10	ff_avg	Double	4	Kecepatan Angin Rata-rata
11	ddd_car	Double	4	Arah Angin Terbanyak

Pada Tabel 4.18 merupakan tabel yang menyimpan dataset klimatologi yang di dapat dari *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman* dari tahun 1985 sampai tahun 2021, hanya saja untuk batasan masalah pada penelitian ini kolom data yang di gunakan hanyalah kolom *rr*.

#### 4.1.3.1.4. *Tabel Proyeksi Riwayat*

Merupakan tabel yang di gunakan untuk menampung hasil proyeksi yang nantinya dapat di gunakan untuk menjad perbandingan pada Proyeksi lainnya dengan rancangan struktur tabel seperti pada Tabel 4.18 berikut :

- Nama *Database* : proyeksi
- Nama Tabel : proyeksi\_riwayat

**Tabel 4.19. Tabel *proyeksi\_riwayat***

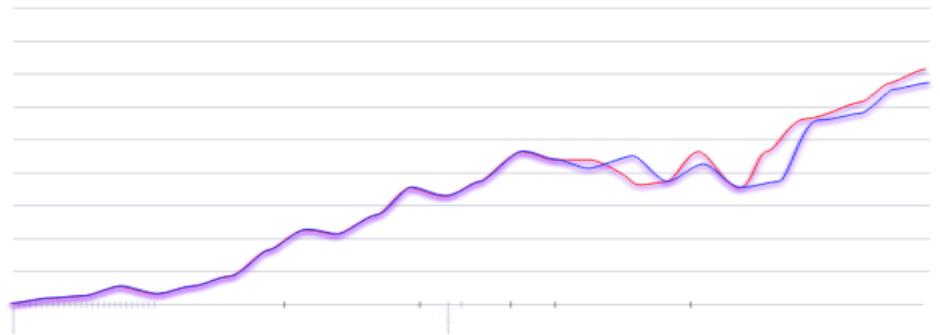
No	Field Name	Type	Width	Description
1	id	Integer	11	Primary Key
2	<i>timestep</i>	Integer	11	Panjang Sequence/Timestep
3	max_batch_size	Integer	11	Ukuran Batch Maksimal
4	max_epoch	Integer	11	Maksimal Epoch

5	<i>layer_size</i>	Integer	11	Jumlah Hidden <i>Layer</i>
6	<i>unit_size</i>	Integer	11	Jumlah <i>Unit</i> Setiap <i>Layer</i>
7	dropout	Double	8	Probabilitas Dropout
8	<i>learning_rate</i>	Double	8	Nilai <i>Learning Rate</i>
9	row_start	Integer	11	ID Awal Baris Data
10	row_end	Integer	11	ID Akhir Baris Data
11	num_predict	Integer	11	Jumlah Prediksi ke Depan
12	logs	Text	-	Catatan <i>Output</i>
13	hdf	Blob	-	H5 tempat bobot dan <i>bias</i>

Pada Tabel 4.19 berperan penting dalam menyimpan Riwayat setiap Proyeksi di lakukan, semua nilai *hyperparameter* dan *parameter* yg diperlukan di simpan ke dalam tabel tersebut ketika selesai melakukan Proyeksi, guna agar dapat di buka Kembali Proyeksi yang telah di lakukan untuk menjadi pembanding dari Proyeksi selanjutnya.

#### 4.1.3.2. Analisa Hasil *Output* Pengujian

*Output* dari hasil perhitungan model LSTM adalah bentuk tabel dan grafik statistik yang berisi hasil akurasi dari data aktual dengan data prediksi, serta hasil prediksi hari kedepanya seperti yang terlihat pada Gambar 4.3 berikut.



**Gambar 4.3. Contoh Sketsa Grafik *Output Hasil Pengujian***

Pada Gambar 4.3 Hasil dari algoritma LSTM akan di tampilkan dari proses tersebut guna mengetahui akurasi hasil prediksi, bentuk hasil prediksi akan berbentuk grafik garis dimana terdapat 3 garis yaitu garis hasil proyeksi, garis data sebenarnya, dan garis prediksi masa depan.

## 4.2. Perancangan

Perancangan merupakan proses di mana suatu sistem di gambarkan sesuai kebutuhan pada fase analisis. Tahap yang di lakukan dalam perancangan sesuai dengan arsitektur sistem yang di perlukan agar setiap agar setiap sistem yang di bangun memiliki konstruksi yang baik, guna mempermudah untuk melakukan pengembangan apabila di perlukan nantinya di lain waktu.

### 4.2.1. Perancangan Model

Pada perancangan model dilakukan pengumpulan beberapa fakta kebutuhan yang mendukung dalam arsitektur rancangan sistem. Dengan menggunakan *Unified Modelling Language* (UML) sebagai *tools* dalam memaparkan alur arsitektur dari sistem yang di rancang. Adapun UML yang digunakan adalah sebagai berikut :

#### **4.2.1.1. Use case Diagram**

*Use case Diagram* menggambarkan bagaimana pengguna atau aktor menggunakan atau memanfaatkan sistem. Pada *use case* diagram mendeskripsikan interaksi dari beberapa aktor dengan sistem yang dirancang.

##### **4.2.1.1.1. Definisi Aktor**

Definisi aktor adalah aktivitas yang bisa dilakukan oleh para pengguna dalam menggunakan sistem. Definisinya dapat dijelaskan pada Tabel 4.20 berikut :

**Tabel 4.20. Definisi Aktor**

No	Aktor	Deskripsi
1	<i>Admin</i>	Aktor <i>Admin</i> melakukan operasi dan pengelolaan terhadap sistem. Pengelolaan tersebut dilakukan dengan akses secara penuh terhadap sistem, baik berupa <i>Create (tambah data)</i> , <i>Read (daca data)</i> <i>Update (edit data)</i> dan <i>Delete (hapus data)</i> data yang berhubungan dengan sistem.

Pada Gambar 4.20 menjelaskan seluruh aktor yang terlibat dalam sistem di jelaskan, tentang hak akses dan fitur yang dapat digunakan oleh aktor tersebut di dalam sistem yang di rancang.

##### **4.2.1.1.2. Definisi Use case**

Definisi *use case* adalah kegiatan yang dapat dilakukan oleh aktor di dalam sebuah sistem. Definisinya dapat dijelaskan pada Tabel 4.21 berikut :

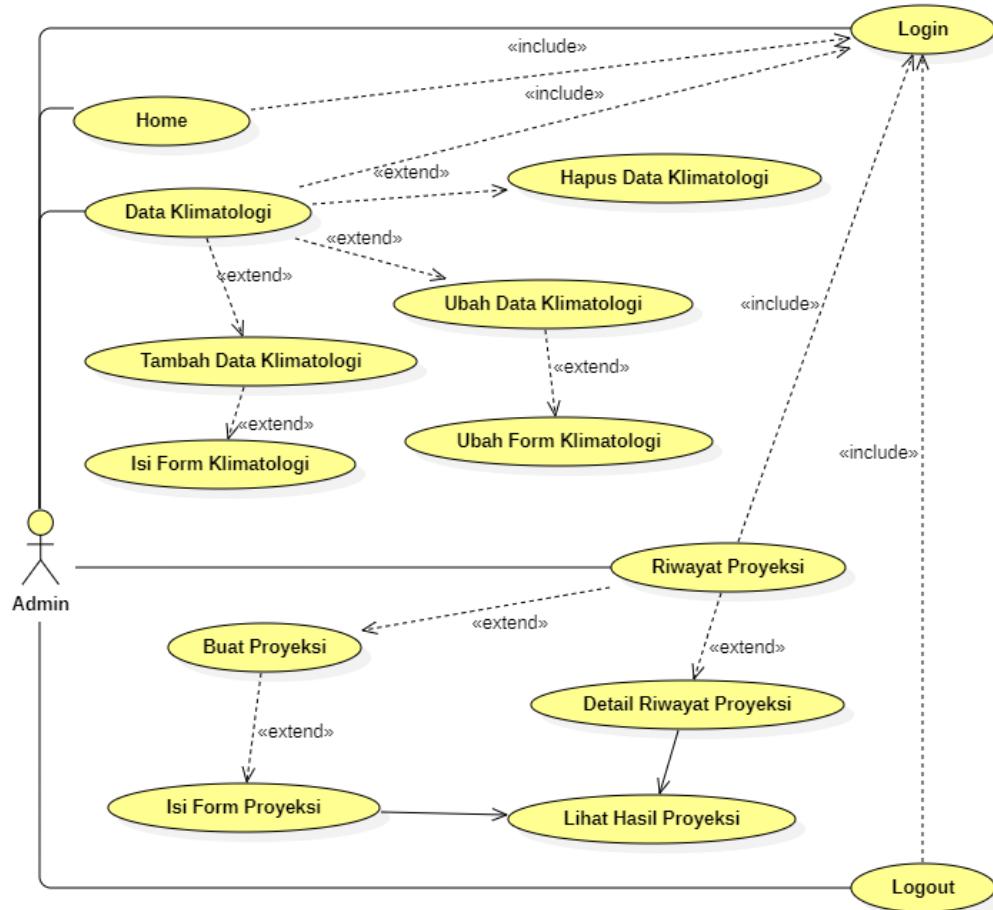
**Tabel 4.21. Definisi Use case yang digunakan**

No	Use case	Aktor	Deskripsi
----	----------	-------	-----------

1	<i>Home</i>	<i>Admin</i>	Merupakan <i>Use case</i> untuk menampilkan halaman Utama yang menampilkan, keterangan dari sistem.
2	<i>Login</i>	<i>Admin</i>	<i>Use case</i> yang di gunakan aktor untuk authentikasi ke dalam sistem untuk mendapatkan hak akses terhadap beberapa <i>use case</i> di dalam sistem.
3	<i>Logout</i>	<i>Admin</i>	<i>Use case</i> yang digunakan untuk melepaskan authentikasi <i>login</i> pada sistem, menghapus sesi <i>login</i> dan hak akses yang berjalan.
4	<i>Data Klimatologi</i>	<i>Admin</i>	Merupakan <i>Use case</i> untuk menampilkan kumpulan data klimatologi yang ada di sistem.
5	<i>Tambah Data Klimatologi</i>	<i>Admin</i>	<i>Use case</i> yang digunakan apabila <i>admin</i> ingin menambah baris baru pada kumpulan data klimatologi
6	<i>Isi Form Klimatologi</i>	<i>Admin</i>	<i>Use case</i> dimana <i>admin</i> mengisi data klimatologi yang akan di <i>input</i> ke sistem.
7	<i>Ubah Data Klimatologi</i>	<i>Admin</i>	<i>Use case</i> yang digunakan apabila <i>admin</i> ingin mengubah baris suatu data pada kumpulan data klimatologi
8	<i>Ubah Form Klimatologi</i>	<i>Admin</i>	<i>Use case</i> dimana <i>admin</i> mengubah isi <i>form</i> dari data klimatologi yang akan di ubah.
9	<i>Hapus Data Klimatologi</i>	<i>Admin</i>	<i>Use case</i> yang digunakan apabila <i>admin</i> ingin menghapus baris suatu data pada kumpulan data klimatologi

10	<i>Riwayat Proyeksi</i>	<i>Admin</i>	<i>Use case</i> yang berguna untuk melihat Riwayat Proyeksi yang telah dilakukan sebelumnya.
11	<i>Buat Proyeksi</i>	<i>Admin</i>	<i>Use case</i> yang berguna untuk melakukan Proyeksi baru dari data Klimatologi dengan metode LSTM.
12	<i>Isi Form Proyeksi</i>	<i>Admin</i>	<i>Use case</i> dimana <i>admin</i> mengisi beberapa <i>parameter</i> yang harus di <i>inputkan</i> sebelum melakukan Proyeksi
13	<i>Detail Riwayat Proyeksi</i>	<i>Admin</i>	<i>Use case</i> yang digunakan apabila <i>admin</i> ingin melihat hasil Proyeksi dari suatu proyeksi pada kumpulan Riwayat proyeksi
14	<i>Lihat Hasil Proyeksi</i>	<i>Admin</i>	<i>Use case</i> dimana <i>admin</i> dapat melihat hasil proyeksi yang dilakukan.
16	<i>Ubah Form Profil</i>	<i>Admin</i>	<i>Use case</i> dimana <i>admin</i> mengubah isi <i>form</i> profil <i>admin</i> yang sudah ada.
17	<i>Isi Form Password</i>	<i>Admin</i>	<i>Use case</i> dimana <i>admin</i> perlu mengisi password untuk keperluan authentikasi saat ingin mengubah data profil <i>admin</i> .

Berdasarkan Tabel 4.21 untuk rancangan UML berupa *Use case Diagram* dapat dilihat pada Gambar 4.4 di bawah ini :



**Gambar 4.4. Use case Diagram Admin**

Berdasarkan Gambar 4.4 terdapat 5 case inti yaitu case *Login*, *Home*, *Data Klimatologi*, *Riwayat Proyeksi* dan *Logout*, selebihnya adalah extends dari case inti tersebut seperti yang di jelaskan pada Tabel 4.21 sebelumnya.

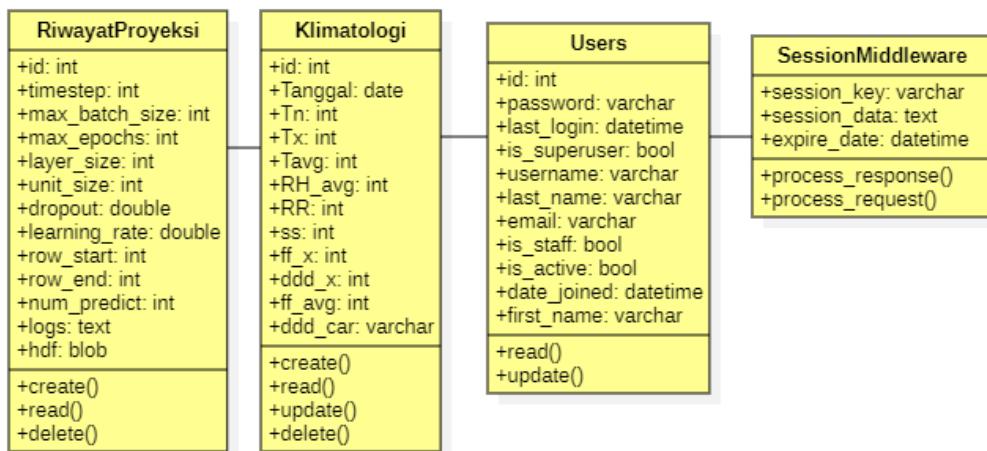
#### 4.2.1.2. Class Diagram

*Class Diagram* menjelaskan mengenai jenis – jenis objek yang terdapat di dalam sebuah sistem dan berbagai hubungan statis yang terdapat pada sistem. Merupakan inti dari pengembangan dan desain dari program berrorientasi objek. Definisi dari *Class Diagram* dapat di lihat pada Tabel 4.22 berikut ini :

**Tabel 4.22. Definisi Class Diagram**

No	Class Diagram	Deskripsi
1	<i>Users</i>	<i>Class Diagram</i> di gunakan untuk <i>Object Relation Mapping (ORM)</i> untuk melakukan query dan manipulasi data <i>users</i> yang tergabung dalam sistem dari <i>database</i> dengan API DMBS MySQL.
2	<i>Klimatologi</i>	<i>Class Diagram</i> di gunakan untuk <i>Object Relation Mapping (ORM)</i> untuk melakukan query dan manipulasi data dari <i>database</i> dengan API DMBS MySQL.
4	<i>RiwayatProyeksi</i>	Merupakan <i>Class Diagram</i> yang berfungsi untuk menyimpan model dari LSTM beserta <i>parameter</i> hasil
5	<i>SessionMiddleware</i>	Merupakan <i>Class Diagram</i> yang berfungsi untuk menyimpan data sesi authentikasi <i>users</i> .

Berdasarkan Tabel 4.22 rancangan UML berupa *Use case Diagram* dapat dilihat pada Gambar 4.5 di bawah ini :



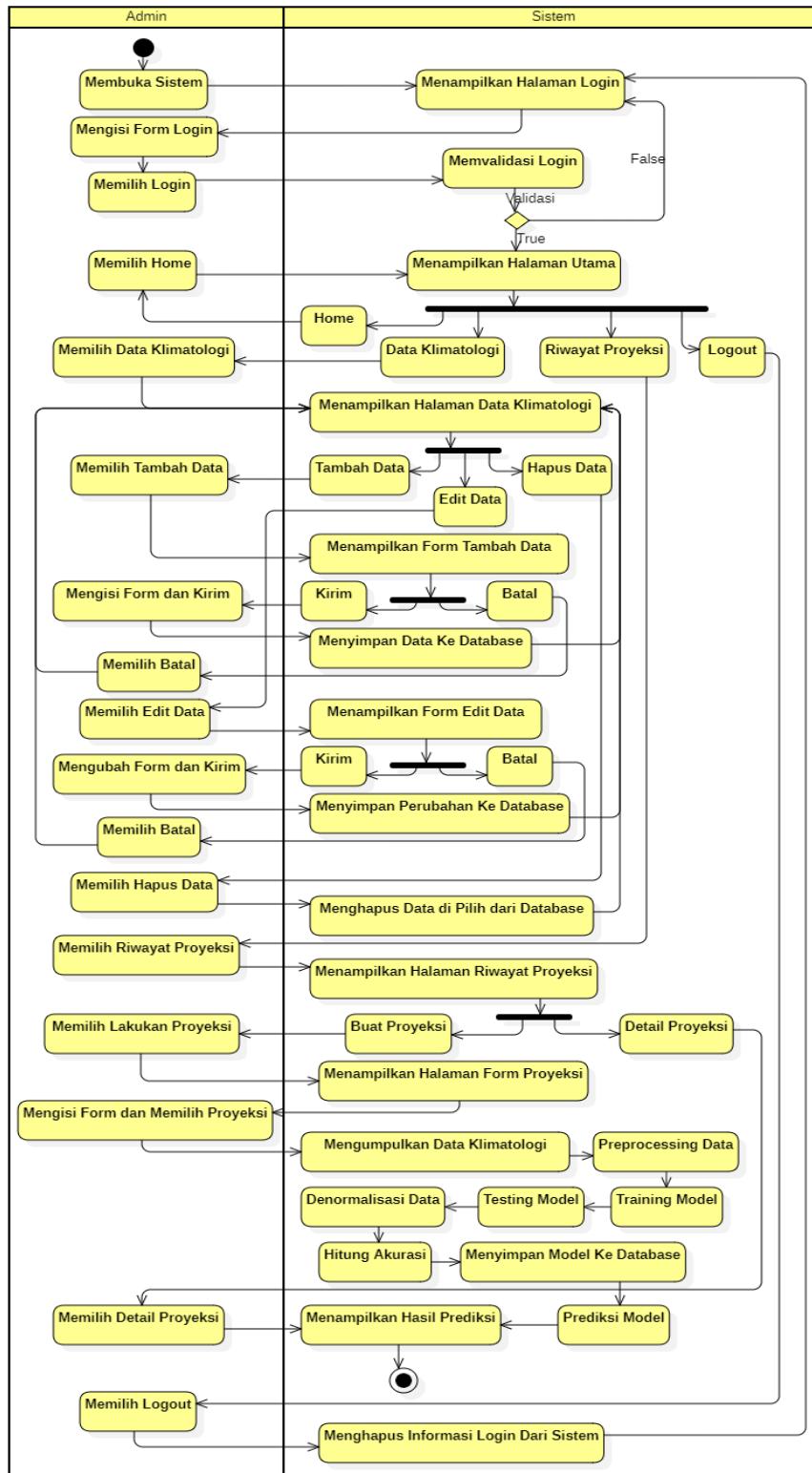
**Gambar 4.5. Class Diagram**

Class diagram Pada Gambar 4.5 seperti *class RiwayatProyeksi* dan *Klimatologi* memiliki keterkaitan pada kolom *row\_start* dan *row\_end* pada *RiwayatProyeksi* dengan *id* pada *Klimatologi*, sebagai penanda rentang data yang akan di gunakan dalam melakukan sebuah Proyeksi.

#### 4.2.1.3. Activity Diagram

*Activity diagram* adalah teknik untuk mendeskripsikan logika prosedural, aliran kerja dalam banyak kasus Activity diagram menggambarkan bagaimana aktivitas yang terjadi dalam sistem yang dirancang. Activity diagram sama seperti halnya flowchart yang menggambarkan proses yang terjadi antara aktor dan sistem.

Pada activity diagram yang di rancang *admin* perlu melakukan authentikasi terlebih dahulu untuk mendapatkan hak akses menggunakan sistem lebih lanjut, Setelah *Admin* melakukan authentikasi *login admin*, *admin* memiliki hak akses untuk masuk ke dalam halaman home, data klimatologi untuk mengelola data Klimatologi, dan proyeksi untuk melakukan proyeksi data Klimatologi, dan halaman edit profil untuk mengelola akun *admin*. seperti yang terlihat pada Gambar 4.6 berikut :



**Gambar 4.6. Activity Diagram Admin**

Berdasarkan Gambar 4.6 perancangan diagram di atas menggambarkan apa saja aktivitas *admin* pada sistem. Mulai dari membuka sistem sampai melihat

halaman utama dari sistem. Pada saat *admin* mengakses sistem dan berhasil tahap validasi setelah melakukan *login admin* langsung di arahkan ke halaman utama yaitu halaman home, dan dapat mengakses beberapa menu lainnya seperti Data Klimatologi, Riwayat Proyeksi, dan *Logout*.

Pada Data Klimatologi *admin* dapat mengelola data klimatologi seperti membaca Data Klimatologi, Tambah Data Klimatologi, Edit Data Klimatologi, dan Hapus Data Klimatologi.

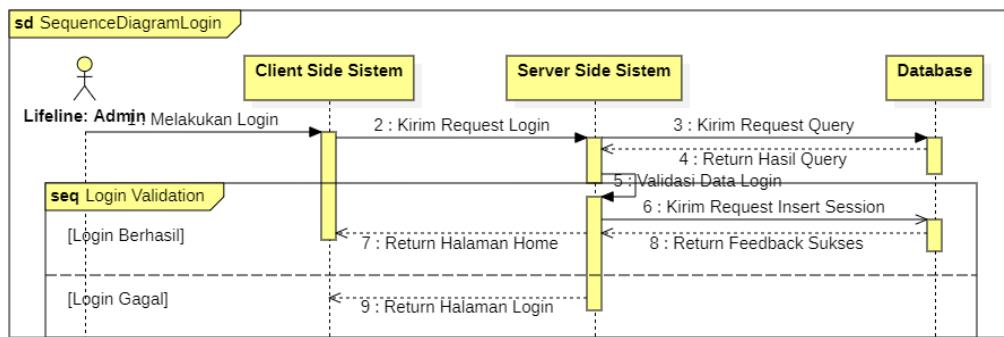
Pada Riwayat Proyeksi melihat Riwayat hasil Proyeksi sebelumnya yang sudah di lakukan berdasarkan *hyperparameter* pendukung Proyeksi, *admin* juga dapat melakukan proyeksi yang baru dengan mengakses Lakukan Proyeksi di mana *admin* harus mengisi beberapa *hyperparameter* pendukung dengan mengisi *form* untuk melakukan Proyeksi data yang nantinya sistem menampilkan hasil Proyeksi berbentuk tabel dan diagram statistik.

#### **4.2.1.4. Sequence Diagram**

*Sequence diagram* merupakan suatu rangkaian yang mendeskripsikan alur kerja dan interaksi-interaksi yang terjadi, dan menjelaskan hubungan timbal balik antara pengguna dan sistem saat berinteraksi. *Sequence diagram* dipengaruhi oleh *use case diagram*, dengan demikian masing-masing *use case* memiliki satu *sequence diagram* yang mendeskripsikan alur kerja dan interaksi yang ada saat *use case* dijelaskan.

#### 4.2.1.4.1. Sequence Diagram Login

*Sequence Diagram* ini menjelaskan urutan yang dilakukan oleh *admin* untuk mendapatkan hak akses sistem dengan melakukan *login* sistem. Dapat di lihat pada Gambar 4.7 berikut ini :

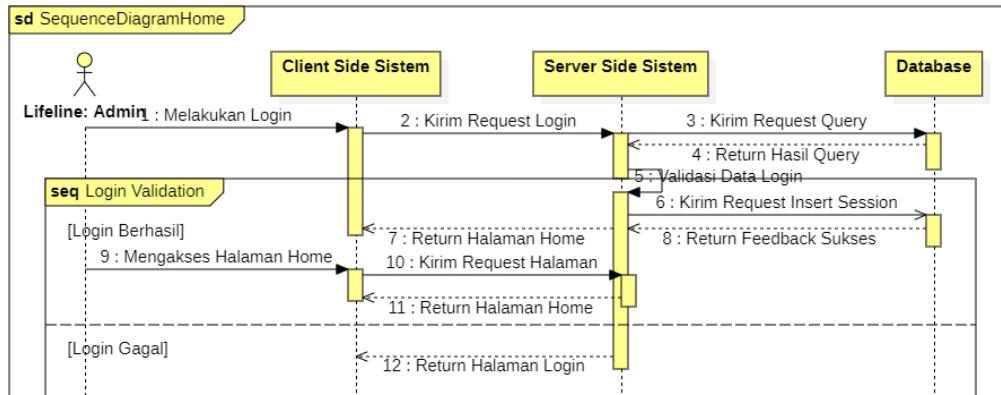


**Gambar 4.7. Sequence Diagram Login**

Sesuai urutan Diagram Gambar 4.7 di atas dapat dilihat bagaimana interaksi pada sistem yang dibangun. Diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di *inputkan* sama dengan *database* maka *admin* dapat hak akses sistem dan sistem untuk menyimpan data sesi *admin* ke dalam sistem dan *database* sebagai penanda kalau *admin* tersebut sedang *login* saat sesi yang berlangsung. Jika data yang di *input form* *login* salah maka *admin* langsung di alihkan kembali ke halaman *login*.

#### 4.2.1.4.2. Sequence Diagram Home

*Sequence Diagram* ini menjelaskan urutan yang dilakukan oleh *admin* agar dapat mengakses halaman home dengan melakukan *login* sistem. Dapat di lihat pada Gambar 4.8 berikut ini :

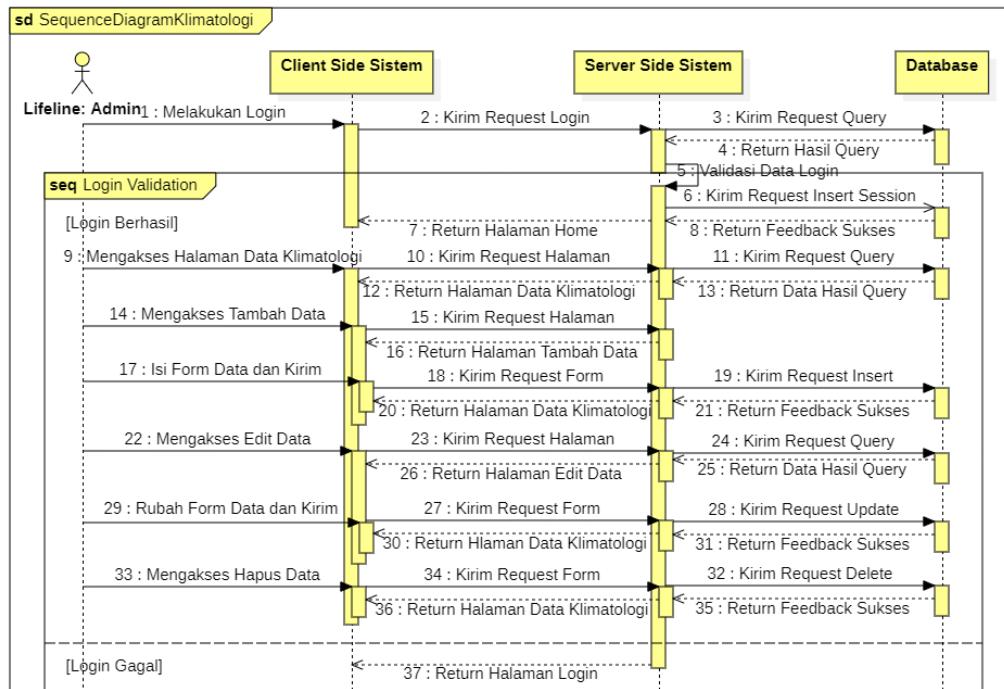


**Gambar 4.8. Sequence Diagram Home**

Sesuai urutan Diagram Gambar 4.8 di atas dapat dilihat bagaimana interaksi pada sistem yang dibangun. Diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di *inputkan* sama dengan *database* maka *admin* dapat mengakses halaman *home*. Jika data yang di *input form login* salah maka *admin* di arahkan kembali ke halaman *login*.

#### 4.2.1.4.3. Sequence Diagram Data Klimatologi

*Sequence Diagram* ini menjelaskan urutan yang dilakukan oleh *admin* agar dapat mengakses halaman data klimatologi dan mengelola data klimatologi tersebut dengan melakukan *login* sistem. Dapat di lihat pada Gambar 4.9 berikut ini :



**Gambar 4.9. Sequence Diagram Data Klimatologi**

Sesuai urutan Diagram Gambar 4.9 di atas dapat dilihat bagaimana interaksi pada sistem yang dibangun. Diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di *inputkan* sama dengan *database* maka *admin* dapat mengakses halaman data Klimatologi dan menggunakan fitur di dalamnya seperti tambah data Klimatologi, Jika data yang di *input form login* salah maka *admin* di arahkan kembali ke halaman *login*.

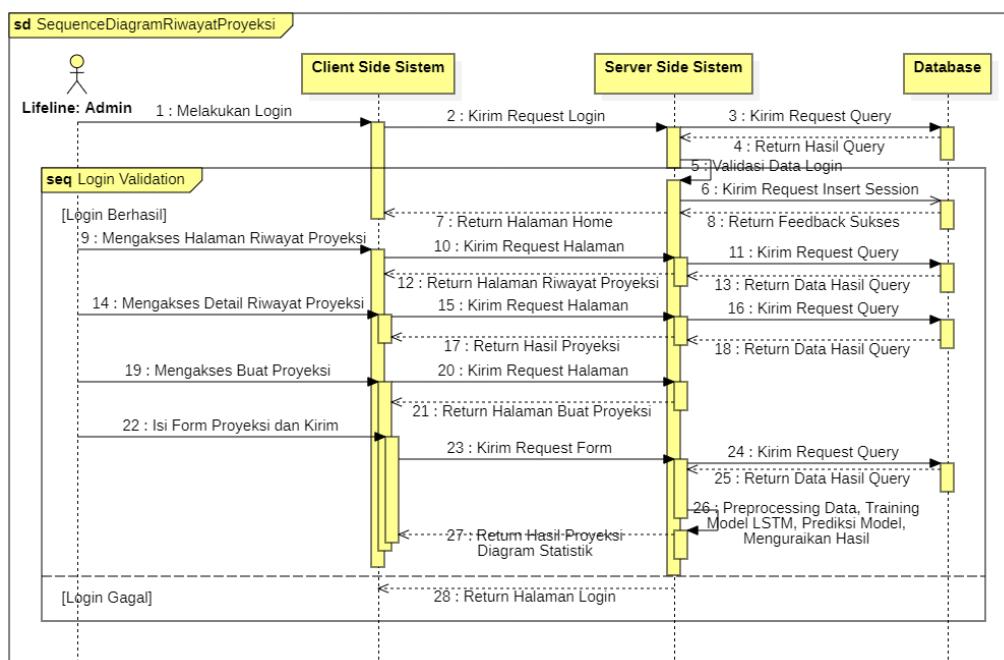
Pada data Klimatologi *admin* dapat melakukan *input form* dan mengirim permintaan ke *server* agar *server* dapat menyimpan data *form* yang di *inputkan* *admin* ke dalam *database* yang nantinya *database* memberikan tanggapan sukses di mana *server* meneruskan pesan sukses tadi dengan mengalihkan *admin* tadi Kembali ke halaman data Klimatologi.

Pada edit data Klimatologi sama halnya dengan Tambah data hanya saja *form input* otomatis terisi dengan data yang ada di *database* dan *admin* di haruskan mengubah *form* tersebut dan mengirim permintaan ke *server*.

Pada hapus data Klimatologi *admin* dapat menghapus data yang di pilih dan mengirim permintaan ke *server*.

#### 4.2.1.4.4. Sequence Diagram Riwayat Proyeksi

*Sequence Diagram* ini menjelaskan urutan yang dilakukan oleh *admin* agar dapat mengakses halaman Riwayat Proyeksi guna melakukan prediksi dari data klimatologi dengan melakukan *login* sistem. Dapat di lihat pada Gambar 4.10 berikut ini :



**Gambar 4.10. Sequence Diagram Riwayat Proyeksi**

Sesuai urutan Diagram Gambar 4.10 di atas dapat dilihat bagaimana interaksi pada sistem yang dibangun. Diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di *input*kan

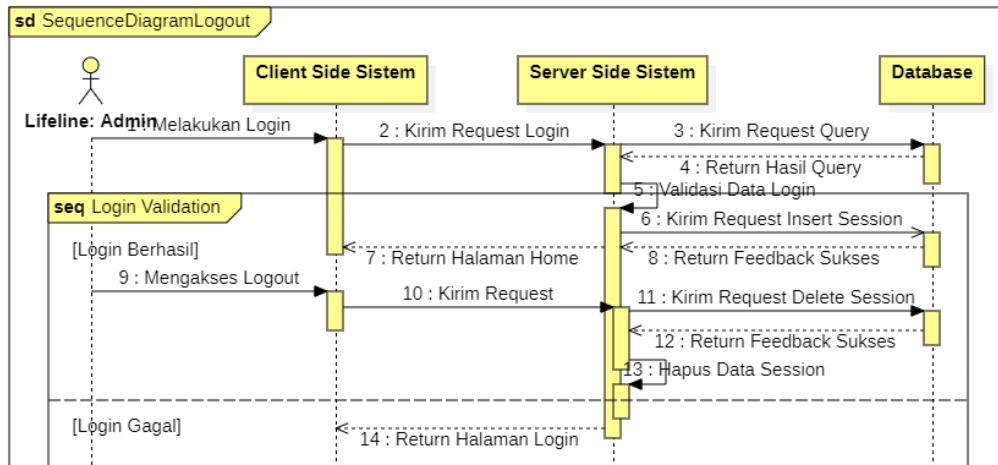
sama dengan *database* maka *admin* dapat mengakses halaman Riwayat Proyeksi, Jika data yang di *input form login* salah maka *admin* di arahkan kembali ke halaman *login*.

Pada halaman Riwayat Proyeksi *admin* dalam melihat detail setiap baris riwayat Proyeksi yang di pilih untuk melihat hasil Proyeksi dari Riwayat Proyeksi sebelumnya.

*Admin* juga dapat membuat baru Proyeksi dengan mengakses halaman buat Proyeksi, selanjutnya *admin* di haruskan untuk mengisi *form* opsi dan *input* untuk menentukan kriteria Proyeksi yang di inginkan, setelah *admin* mengirim permintaan ke sistem maka sistem membaca semua data klimatologi yang di gunakan untuk melakukan Proyeksi dengan tahapnya seperti, *preprocessing* data, *training* model LSTM, prediksi model, dan mengembalikan hasil berupa uraian hasil berbentuk diagram statistik.

#### **4.2.1.4.5. Sequence Diagram Logout**

*Sequence Diagram* ini menjelaskan urutan yang dilakukan oleh *admin* untuk *logout* guna menghapus sesi yang sedang berlangsung dan keluar dari sistem di mana *admin* harus melakukan *login* terlebih dahulu. Dapat di lihat pada Gambar 4.11 berikut ini :



**Gambar 4.11. Sequence Diagram Logout**

Sesuai urutan Diagram Gambar 4.11 di atas dapat dilihat bagaimana interaksi pada sistem yang dibangun. Diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di *inputkan* sama dengan *database* maka *admin* alah berada dalam status sesi *login*. Jika data yang di *input form login* salah maka *admin* di arahkan kembali ke halaman *login*.

*Logout* hanya dapat di lakukan jika *admin* dalam sesi *login*, *admin* dapat keluar dari sistem dengan mengakses *logout* dan permintaan di kirimkan ke *server*, kemudian *server* mengirimkan permintaan ke *database* agar data sesi *admin* yang berlangsung untuk di hapus dari *database* sekaligus sistem menghapus data sesi dari sistem, terakhir sistem mengalihkan *admin* ke halaman *login* kembali.

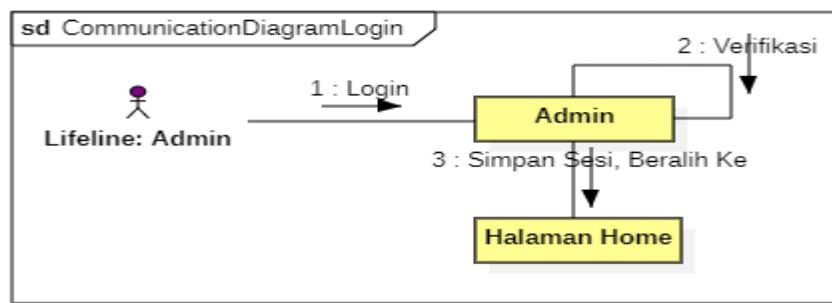
#### 4.2.1.5. Collaboration Diagram

*Collaboratioan Diagram* adalah cara alternatif untuk mengetahui tahap-tahap terjadinya suatu aktivitas. Perbedaan antara *Collaboration Diagram* dan *Squence Diagram* adalah *Collaboration Diagram* memperlihatkan bagaimana hubungan

antara beberapa objek berdasarkan urutan dari pesan, sedangkan *Sequence Diagram* memperlihatkan bagaimana urutan kejadian berdasarkan waktu.

#### **4.2.1.5.1. Collaboration Diagram Login**

*Collaboration Diagram* ini menjelaskan di mana *admin* perlu melakukan *login* dan sukses dalam validasi untuk mendapatkan hak akses *admin* dalam sistem, yang kemudian di alihkan ke halaman home. Dapat di lihat pada Gambar 4.12 berikut.



**Gambar 4.12. Collaboration Diagram Login**

Pada *Collaboration Diagram* Gambar 4.12 terlihat bagaimana kolaborasi actor *admin* dengan sistem yang dibangun, diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di *inputkan* sama dengan *database* maka *admin* dapat hak akses sistem dan sistem untuk menyimpan data sesi *admin* ke dalam sistem dan *database* sebagai penanda kalau *admin* tersebut sedang *login* saat sesi yang berlangsung.

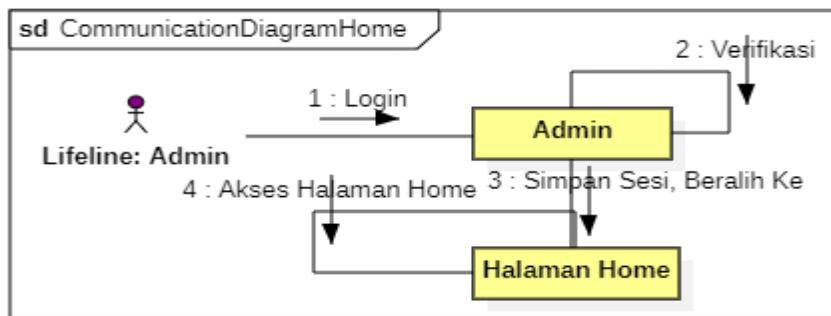
Ketika *admin* sudah *login* *admin* akan di alihkan ke halaman home dan mendapat status dalam keadaan sesi *login*.

#### **4.2.1.5.2. Collaboration Diagram Home**

*Collaboration Diagram* ini menjelaskan di mana setelah *admin* perlu verifikasi bahwa dia adalah *admin* mendapatkan hak akses sebagai *admin*, kemudian *admin*

tersebut dapat berkolaborasi dengan sistem seperti mengakses halaman home.

Dapat di lihat pada Gambar 4.13 berikut.



**Gambar 4.13. Collaboration Diagram Home**

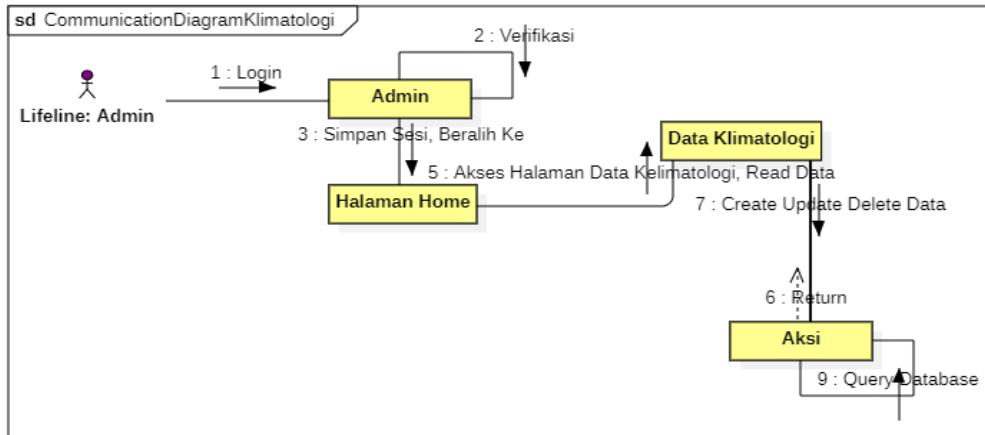
Pada *Collaboration Diagram* Gambar 4.13 terlihat bagaimana kolaborasi actor *admin* dengan sistem yang dibangun, diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di inputkan sama dengan *database* maka *admin* dapat hak akses sistem dan sistem untuk menyimpan data sesi *admin* ke dalam sistem dan *database* sebagai penanda kalau *admin* tersebut sedang *login* saat sesi yang berlangsung.

Ketika *admin* sudah *login* *admin* akan di alihkan ke halaman home dan mendapat status dalam keadaan sesi *login*, kemudian *admin* dapat selalu beriteraksi dengan halaman home dengan mengakses Kembali halaman home selama *admin* masih dalam sesi *login*.

#### **4.2.1.5.3. Collaboration Diagram Data Klimatologi**

*Collaboration Diagram* ini menjelaskan di mana setelah *admin* perlu verifikasi bahwa dia adalah *admin* mendapatkan hak akses sebagai *admin*, kemudian *admin* tersebut dapat mengelola data Klimatologi dengan mengakses halaman data Klimatologi, di mana *admin* dapat berkolaborasi dengan sistem seperti aksi *Create*,

*Update* dan *Delete*, dan aksi selebihnya di tangani oleh sistem. Dapat di lihat pada Gambar 4.14 berikut.



**Gambar 4.14. Collaboration Diagram Data Klimatologi**

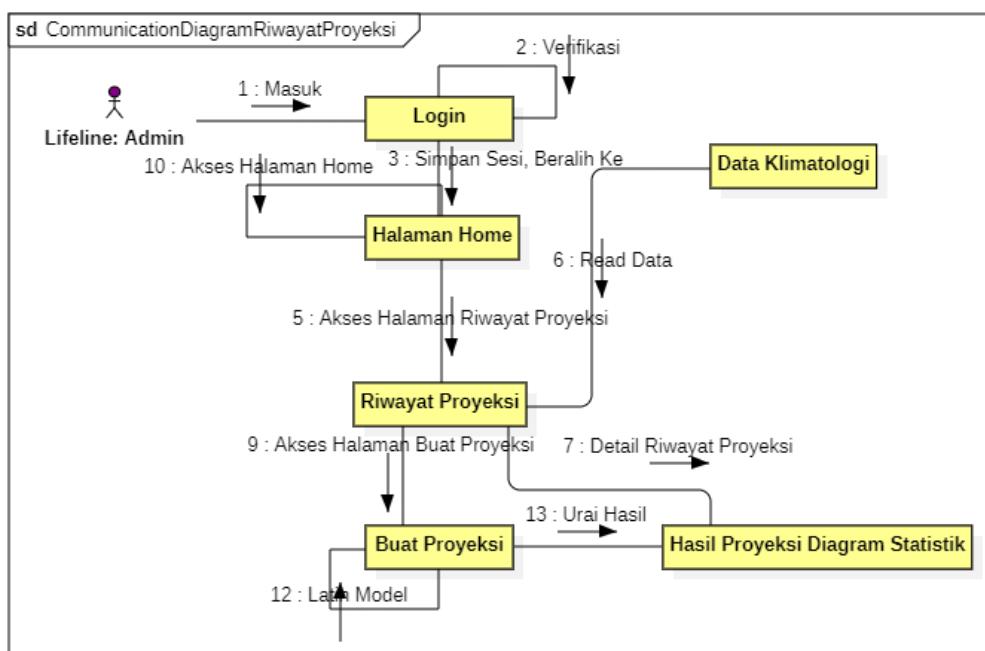
Pada *Collaboration Diagram* Gambar 4.14 terlihat bagaimana kolaborasi actor *admin* dengan sistem yang dibangun, diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di inputkan sama dengan *database* maka *admin* dapat hak akses sistem dan sistem untuk menyimpan data sesi *admin* ke dalam sistem dan *database* sebagai penanda kalau *admin* tersebut sedang *login* saat sesi yang berlangsung.

Ketika *admin* sudah *login* *admin* akan di alihkan ke halaman home, selanjutnya *admin* dapat berkolaborasi dengan data Klimatologi untuk memanajemen data Klimatologi seperti Tambah data edit data dan hapus data klimatologi.

#### 4.2.1.5.4. Collaboration Diagram Riwayat Proyeksi

*Collaboration Diagram* ini menjelaskan di mana setelah *admin* perlu verifikasi bahwa dia adalah *admin* mendapatkan hak akses sebagai *admin*, kemudian *admin* tersebut berkolaborasi dengan melakukan Proyeksi dengan mengakses halaman

Buat Proyeksi di dalam halaman Riwayat Proyeksi, selanjutnya peran sistemlah yang melakukan pengambilan data dan melatih data tersebut dengan model serta menguraikan hasilnya agar mudah di mengerti oleh *admin*. Dapat di lihat pada Gambar 4.15 berikut.



**Gambar 4.15. Collaboration Diagram Riwayat Proyeksi**

Pada *Collaboration Diagram* Gambar 4.15 terlihat bagaimana kolaborasi actor *admin* dengan sistem yang dibangun, diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di *inputkan* sama dengan *database* maka *admin* dapat hak akses sistem dan sistem untuk menyimpan data sesi *admin* ke dalam sistem dan *database* sebagai penanda kalau *admin* tersebut sedang *login* saat sesi yang berlangsung.

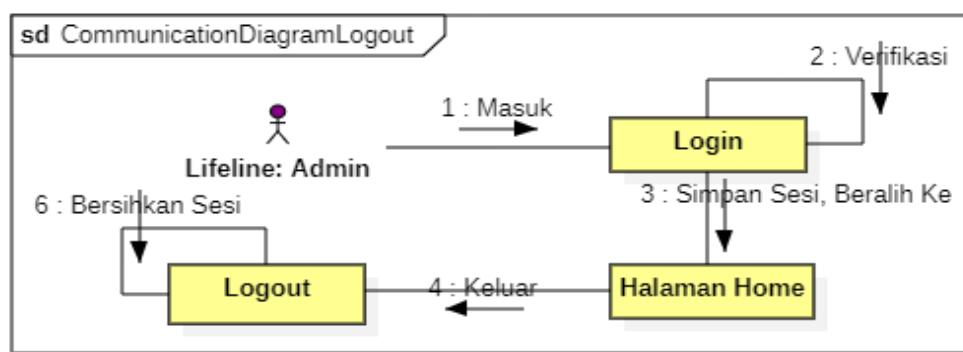
Ketika *admin* sudah *login* *admin* akan di alihkan ke halaman home, selanjutnya *admin* dapat berkolaborasi dengan Riwayat Proyeksi untuk mengakses fitur Proyeksi pada sistem, sistem akan berkolaborasi dengan membaca data table

Klimatologi dari *database* Proyeksi untuk di tampilkan dan dapat di lihat oleh *admin*.

Ketika *admin* sudah di dalam halaman Riwayat Proyeksi *admin* dapat mengakses detail Proyeksi dari baris table Riwayat Proyeksi yang akan di lihat, *admin* juga dapat membuat Proyeksi baru dengan mengakses Buat Proyeksi dimana yg nantinya sistem akan melatih model dan menampilkan hasil Proyeksi berbentuk diagram statistik.

#### **4.2.1.5.5. Collaboration Diagram Logout**

*Collaboration Diagram* ini menjelaskan di mana setelah *admin* perlu verifikasi bahwa dia adalah *admin* mendapatkan hak akses sebagai *admin*, kemudian *admin* tersebut dapat memutuskan kolaborasi dengan sistem atau keluar dari sistem seperti melakukan *logout*, dan sistem berkolaborasi seperti membersihkan data sesi pada *admin* yang masih tersimpan. Dapat di lihat pada Gambar 4.16 berikut.



**Gambar 4.16. Collaboration Diagram Logout**

Pada *Collaboration Diagram* Gambar 4.16 terlihat bagaimana kolaborasi actor *admin* dengan sistem yang dibangun, diawali *admin* harus melalui tahap *login* ke dalam sistem dan sistem melakukan validasi *form* yang di kirim, jika data yang di inputkan sama dengan *database* maka *admin* dapat hak akses sistem dan sistem

untuk menyimpan data sesi *admin* ke dalam sistem dan *database* sebagai penanda kalau *admin* tersebut sedang *login* saat sesi yang berlangsung.

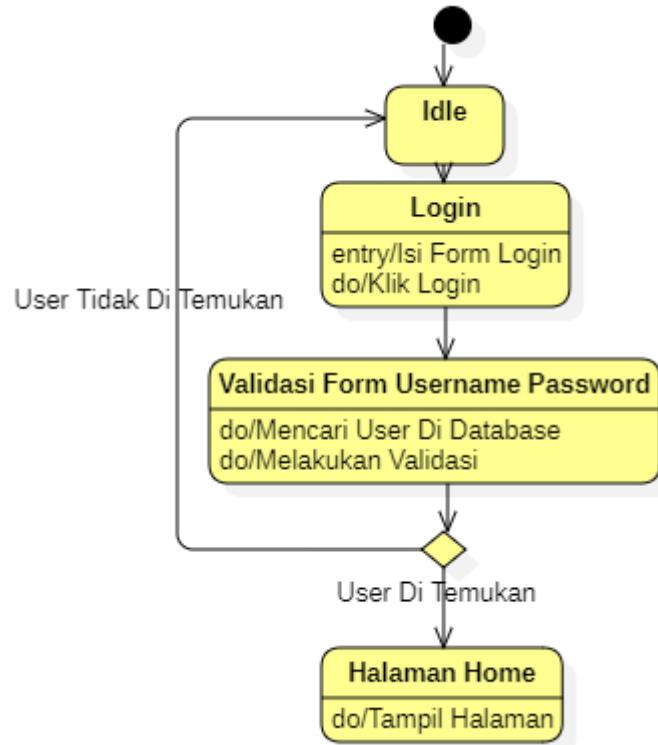
*Logout* hanya dapat di lakukan jika *admin* dalam sesi *login*, *admin* dapat berkolaborasi dengan mengakses *logout* dan permintaan di kirimkan ke *server*, kemudian *server* akan berkolaborasi dengan mengirimkan permintaan ke *database* agar data sesi *admin* yang berlangsung untuk di hapus dari *database* sekaligus sistem menghapus data sesi dari sistem.

#### **4.2.1.6. Statechart Diagram**

*Statechart diagram* menggambarkan perubahan status yang terjadi ketika sistem dijalankan. Perubahan yang terjadi pada suatu objek digambarkan oleh diagram ini dalam bentuk grafik berarah.

##### **4.2.1.6.1. Statechart Diagram Login**

*Statechart Diagram* ini memperlihatkan bagaimana alur perpindahan status *admin* dalam menggunakan sistem mulai dari authentikasi *login* sistem. Dapat di lihat pada Gambar 4.17 berikut.

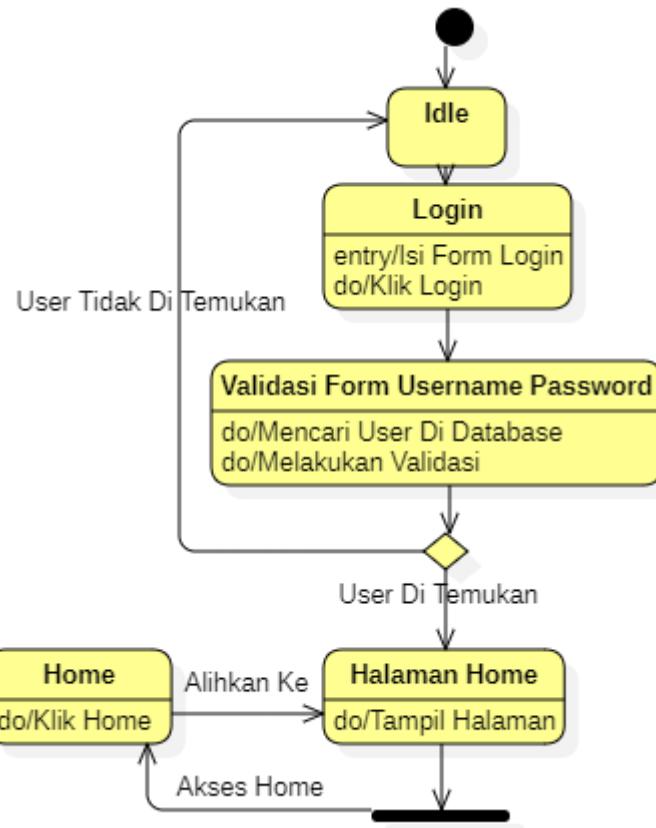


**Gambar 4.17. Statechart Diagram Login**

Dari *statechart* diagram pada Gambar 4.17 di atas dapat di lihat bagaimana alur perpindahan status pada *admin* saat *idle* hingga ke halaman home, di mana *admin* perlu melakukan entri *form* pada *login* dan klik *login*, kemudian sistem mencari *user* di *database* dan melakukan validasi, apabila *user* di temukan maka *admin* di teruskan ke halaman *admin*, apabila *user* tidak di temukan maka *admin* di alihkan kembali ke kondisi sebelumnya.

#### 4.2.1.6.2. Statechart Diagram Home

*Statechart Diagram* ini memperlihatkan bagaimana alur perpindahan status *admin* dalam menggunakan sistem mulai dari authentikasi hingga klik halaman home. Dapat di lihat pada Gambar 4.18 berikut ini :

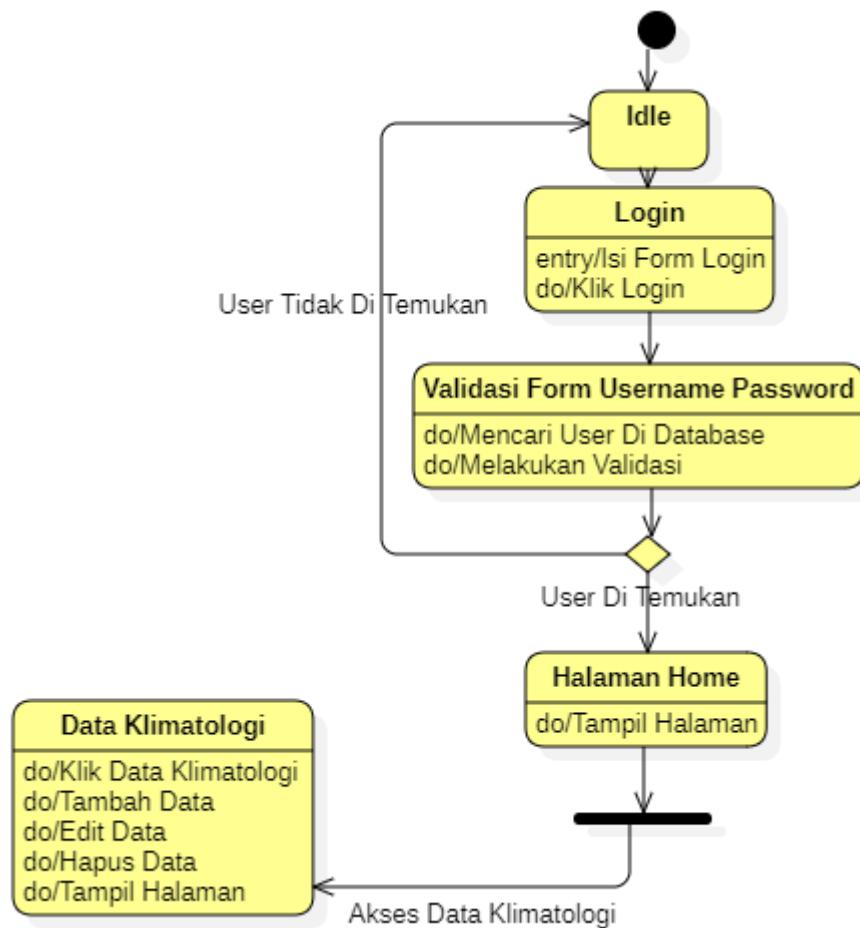


**Gambar 4.18. Statechart Diagram Home**

Dari *statechart* diagram pada Gambar 4.18 di atas Ketika *admin* sudah di verifikasi dan *admin* dalam keadaan status sebagai *admin* dapat mengeklik home agar di alihkan ke halaman home.

#### 4.2.1.6.3. Statechart Diagram Data Klimatologi

*Statechart Diagram* ini memperlihatkan bagaimana alur perpindahan status *admin* dalam menggunakan sistem mulai dari authentikasi *login* hingga akses data klimatologi. Dapat di lihat pada Gambar 4.19 berikut ini :

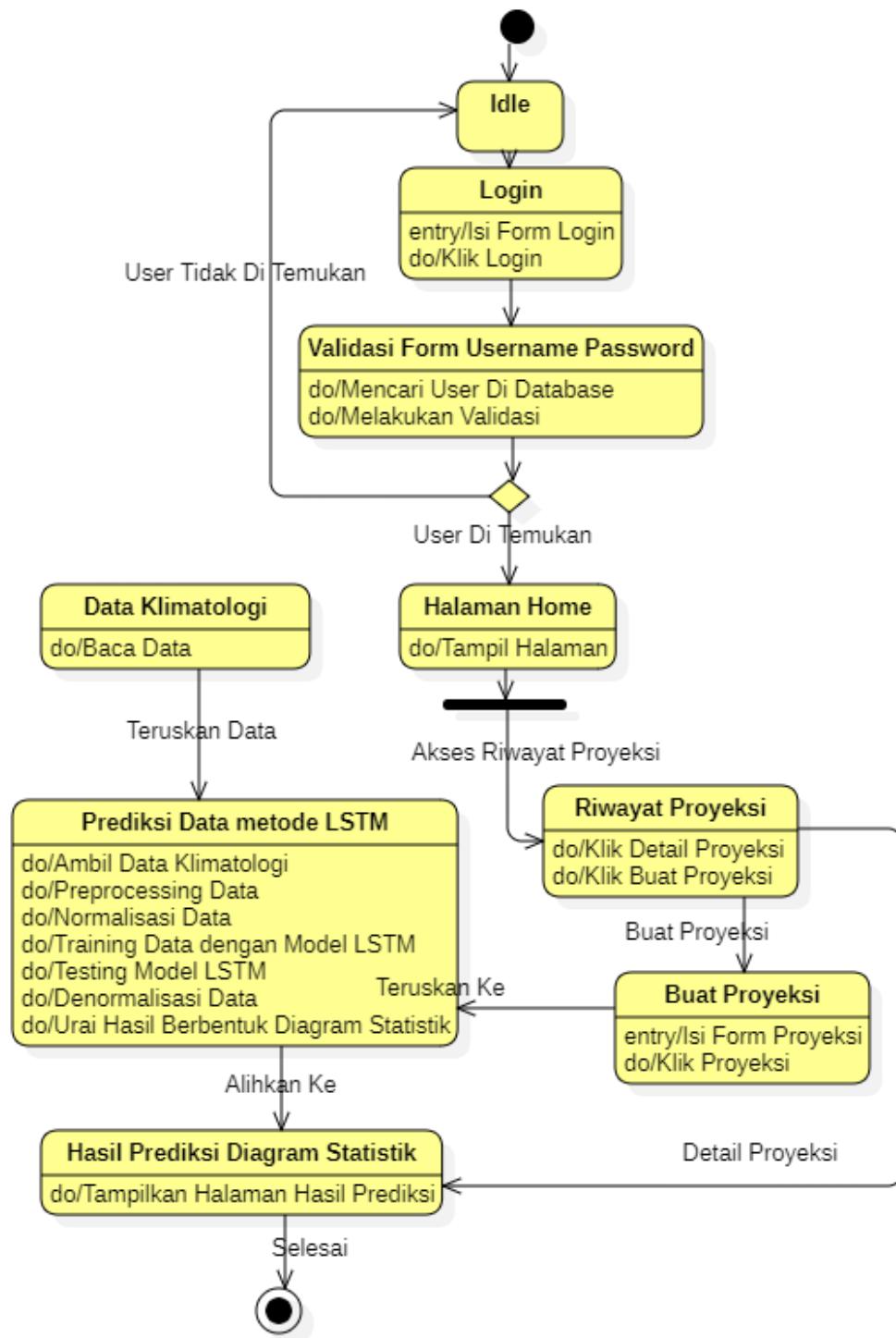


**Gambar 4.19. Statechart Diagram Data Klimatologi**

Dari *statechart* diagram pada Gambar 4.19 di atas Ketika *admin* sudah di verifikasi dan *admin* dalam keadaan status sebagai *admin* dapat mengakses Data Klimatologi, kemudian *admin* dapat melakukan operasi klik Data Klimatologi, Tambah data, Edit Data, Hapus Data.

#### 4.2.1.6.4. Statechart Diagram Riwayat Proyeksi

*Statechart Diagram* ini memperlihatkan bagaimana alur perpindahan status *admin* dalam menggunakan sistem hingga melakukan Proyeksi data. Dapat di lihat pada Gambar 4.20 berikut ini :



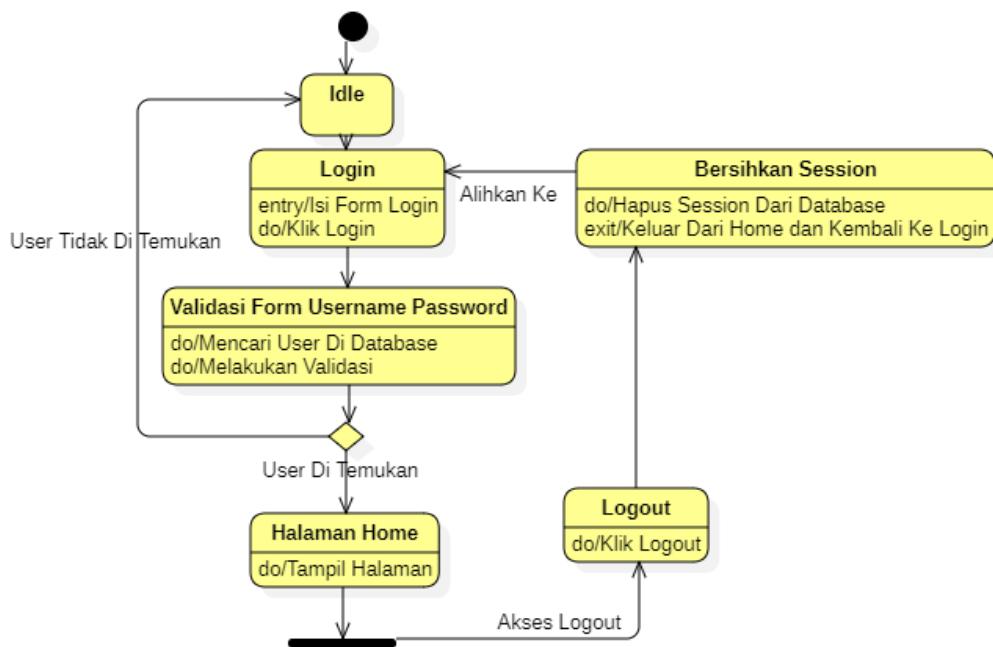
**Gambar 4.20. Statechart Diagram Riwayat Proyeksi**

Dari *statechart* diagram pada Gambar 4.20 di atas Ketika *admin* sudah di verifikasi dan *admin* dalam keadaan status sebagai *admin* dapat langsung

menggunakan fitur Riwayat Proyeksi yang di buat khusus pada penelitian ini. *admin* yang sudah mengakses halaman buat proyeksi dapat mengisi *form* Proyeksi kemudian klik Proyeksi dan *state* selanjutnya diteruskan oleh sistem, pertama-tama sistem mengambil semua data klimatologi, *state* berikutnya sistem melakukan *preprocessing* pada data yang sudah di ambil, lalu melakukan normalisasi data, *state* berikutnya melakukan *training* data dengan model LSTM, lalu data yang sudah di latih kemudian di *testing*, *state* selanjutnya adalah untuk mengembalikan data ke nilai sebelumnya di lakukan denormalisasi data, *state* terakhir sistem menguraikan hasil dan menampilkan hasil berbentuk diagram statistik.

#### 4.2.1.6.5. Statechart Diagram Logout

*Statechart Diagram* ini memperlihatkan bagaimana alur perpindahan status *admin* dalam menggunakan sistem dari melakukan authentikasi hingga melakukan *logout* sistem. Dapat di lihat pada Gambar 4.21 berikut ini :

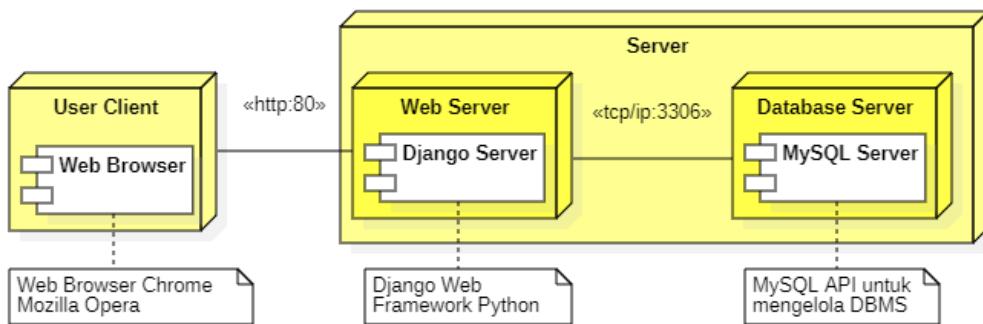


Gambar 4.21. *Statechart Diagram Logout*

Dari *statechart* diagram pada Gambar 4.21 di atas Ketika *admin* sudah di verifikasi dan *admin* dalam keadaan status sebagai *admin* dapat mengeklik *logout* untuk keluar dari *state login*, kemudian sistem menghapus sesi dan sistem dan *database* dan mengembalikan *admin* ke halaman *login*.

#### **4.2.1.7. Deployment Diagram**

*Deployment diagram* menggambarkan detail bagaimana komponen dikembangkan dalam infrastruktur sistem, termasuk di mana komponen terletak protokol jaringan saling berkomunikasi satu sama lain, misalnya seperti tcp/ip dan protokol http, spesifikasi *server*, dan hal-hal lain yang bersifat fisik.



**Gambar 4.22. Deployment Diagram**

Pada Gambar 4.22 terlihat dari *admin* sebagai *user client* menggunakan web browser seperti chrome, mozilla, atau opera untuk berkomunikasi dengan *server* khususnya pada web *server* Django dengan protokol http dengan Port 80, dan web *server* tersebut juga berkomunikasi dengan *database server* khususnya API DBMS MySQL pada protokol jaringan tcp/ip Port 3306, dalam 1 lingkup *server* induk.

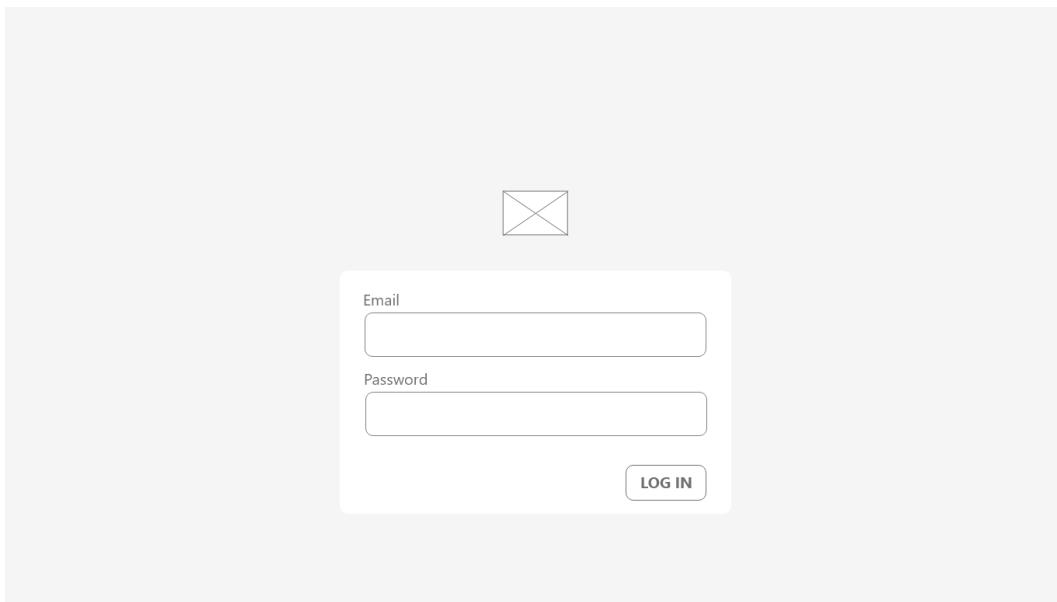
#### **4.2.2. Perancangan Interface**

Desain *Interface* dilakukan dengan tujuan memberikan gambaran dari tampilan yang dilihat oleh *admin* pada saat menggunakan sistem, dan membangun aplikasi

dengan memenuhi prinsip perancangan *interface* yang baik untuk *admin*. Dengan menggunakan *Adobe XD 2021* sebagai tools dalam sebagai rancangan desain sistem yang dibuat nantinya. Berikut adalah beberapa desain *interface* sistem Proyeksi pada penelitian ini :

#### 4.2.2.1. Desain Halaman *Login*

Halaman *login* adalah tampilan awal kinerja proses di mana *admin* harus melakukan authentikasi terlebih dahulu sebelum memasuki sistem setelah *admin login* maka sistem mengarah ke halaman home *admin*. Seperti yang terlihat pada rancangan desain pada Gambar 4.23 berikut :

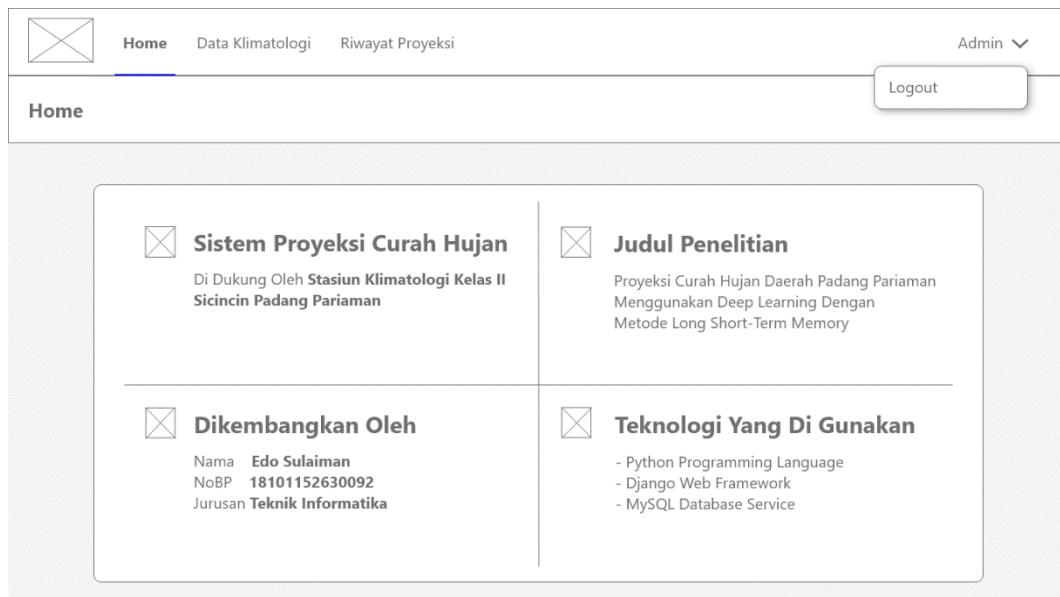


**Gambar 4.23. Desain Halaman *Login***

Pada Gambar 4.23 di atas terdapat *form input* email dan password untuk authentikasi *admin* agar sistem dapat mengirimkan informasi data *login* ke *server*.

#### 4.2.2.2. Desain Halaman Home

Setelah *admin* lolos pada tahap autentikasi *admin* masuk pada halaman home sebagai landasan halaman untuk menelusuri halaman sistem lainnya. Seperti yang terlihat pada rancangan desain pada Gambar 4.24 berikut :



**Gambar 4.24. Desain Halaman Home**

Pada Gambar 4.24 halaman home merupakan halaman landasan yang memperlihatkan tentang sistem yang di gunakan, terdapat tombol *logout* di opsi *dropdown* pada sudut kanan atas apabila *admin* ingin keluar dan mengakhiri sesi akses sistem yang sedang berlangsung.

#### 4.2.2.3. Desain Daftar Tabel Data Klimatologi

Halaman data Klimatologi berbentuk daftar tabel dari data Klimatologi, di mana *admin* dapat melihat potongan-potongan dari keseluruhan data berbentuk tabel. Seperti yang terlihat pada rancangan desain pada Gambar 4.25 berikut :

The screenshot shows a web-based application interface for managing climate data. At the top, there is a navigation bar with links for Home, Data Klimatologi (which is currently selected), and Riwayat Proyeksi. On the right side of the top bar, there are Admin and Logout buttons. Below the navigation bar, the main content area has a title "Data Klimatologi" and a "Tambah Data" button. A search bar labeled "Search:" is located at the top right of the data grid. The data is presented in a table with the following columns:

No	Tanggal	Tn	Tx	Tavg	RH_avg	RR	ss	ff_x	ddd_x	ff_avg	ddd_car	Aksi
int	Date	double	varchar(3)	Button								

**Gambar 4.25. Desain Daftar Tabel Data Klimatologi**

Pada Gambar 4.25 terdapat tombol Tambah data apabila *admin* ingin melakukan penambahan data klimatologi, kemudian juga terdapat *input* search dan show option yang dapat membantu *admin* dalam mencari data yang di kelola dan terdapat kolom aksi agar *admin* dapat menghapus atau beubah data dari baris tabel yang bersangkutan.

#### 4.2.2.4. Desain *Form* Tambah Data Klimatologi

*Admin* langsung berpindah ke halaman Tambah data Klimatologi apabila *admin* melakukan aksi klik tombol Tambah data pada halaman daftar tabel data Klimatologi seperti yang terlihat pada Gambar 4.25. Pada halaman Tambah data inilah *admin* dapat mengisi informasi data yang di tambahkan. Seperti yang terlihat pada rancangan desain pada Gambar 4.26 berikut :

Tambah Data Klimatologi

Tanggal	<input type="text" value="dd/mm/yyyy"/> <input type="button" value="..."/>	Kecepatan Angin Max.	<input type="text"/>
Temperatur Min.	<input type="text"/>	Arah Angin Max.	<input type="text"/>
Temperatur Max.	<input type="text"/>	Kecepatan Angin Mean.	<input type="text"/>
Temperatur Mean.	<input type="text"/>	Arah Angin Terbanyak	<input type="text"/>
Kelembapan Mean.	<input type="text"/>		
Curah Hujan	<input type="text"/>		
Lama Sinar Matahari	<input type="text"/>		

**Kirim**

**Gambar 4.26. Desain *Form* Tambah Data Klimatologi**

Pada Gambar 4.26 terdapat *input form* tanggal bertipe *datepicker* dan selainnya merupakan *input form text* dan terdapat 1 tombol kirim untuk mengirimkan informasi permintaan Tambah data ke *server*.

#### 4.2.2.5. Desain *Form* Edit Data Klimatologi

Apabila *admin* mengakses tombol edit data pada kolom aksi pada tabel yang terdapat pada Gambar 4.25 *admin* langsung di alihkan ke halaman edit data Klimatologi. Seperti yang terlihat pada rancangan desain pada Gambar 4.27 berikut :

The screenshot shows a web-based application interface for managing climate data. At the top, there's a navigation bar with links for Home, Data Klimatologi (which is highlighted in blue), and Riwayat Proyeksi. On the right side of the top bar, there are 'Admin' dropdown and 'Logout' buttons. Below the navigation, the main content area has a header 'Data Klimatologi' and a 'Tambah Data' button. The central part of the screen is titled 'Edit Data Klimatologi'. It contains a table-like structure with two columns. The left column lists parameters: Tanggal, Temperatur Min., Temperatur Max., Temperatur Mean., Kelembapan Mean., Curah Hujan, and Lama Sinar Matahari. The right column lists their corresponding values: 19/06/1998, 22.0, 32.0, 25.4, 88.0, 0.0, and 7.4. To the right of these values are two more columns: 'Kecepatan Angin Max.' (with value 270.0) and 'Arah Angin Max.' (with value W). A large 'Kirim' button is located at the bottom right of the form area.

Tanggal	19/06/1998	Kecepatan Angin Max.
Temperatur Min.	22.0	Arah Angin Max.
Temperatur Max.	32.0	Kecepatan Angin Mean.
Temperatur Mean.	25.4	Arah Angin Terbanyak
Kelembapan Mean.	88.0	
Curah Hujan	0.0	
Lama Sinar Matahari	7.4	

**Gambar 4.27. Desain *Form Edit Data Klimatologi***

Pada Gambar 4.27 *form input text* dan *datepicker* secara otomatis terisi sesuai dengan data dari tombol edit yang di akses pada daftar tabel data Klimatologi sebelumnya.

#### 4.2.2.6. Desain Tabel Riwayat Proyeksi

Halaman Proyeksi berbentuk *form input*, di mana *admin* dapat membuat opsi untuk prediksi yang di lakukan oleh sistem. Seperti yang terlihat pada rancangan desain pada Gambar 4.28 berikut :

The screenshot displays a web-based administrative interface. At the top, there is a navigation bar with links for Home, Data Klimatologi, Riwayat Proyeksi (which is underlined in blue), Admin, and Logout. Below the navigation bar, there are two buttons: 'Proyeksi' and 'Buat Proyeksi'. The main content area features a table titled 'Riwayat Proyeksi'. The table includes a header row with column labels: No, Range, Epoch, Batch, TimeStep, Layer Size, Unit Size, Dropout, Learning Rate, RMSE, Predict, and Aksi. The first data row is shown with the following values: int, text, int, int, int, int, int, double, double, double, double, and Button. Above the table, there are search and show options: 'Show 10 entries' and 'Search: [input field]'. The entire interface is presented within a light gray box.

No	Range	Epoch	Batch	TimeStep	Layer Size	Unit Size	Dropout	Learning Rate	RMSE	Predict	Aksi
int	text	int	int	int	int	int	double	double	double	double	Button

**Gambar 4.28. Desain Tabel Riwayat Proyeksi**

Pada Gambar 4.28 terdapat tombol Buat Proyeksi apabila admin ingin melakukan Proyeksi baru data klimatologi, kemudian juga terdapat input search dan show option yang dapat membantu admin dalam mencari data yang di kelola dan terdapat kolom aksi agar admin memilih detail dari bari data Riwayat Proyeksi yang di pilih.

#### 4.2.2.7. Desain *Form* Buat Proyeksi

Halaman Proyeksi berbentuk *form input*, di mana *admin* dapat membuat opsi untuk prediksi yang di lakukan oleh sistem. Seperti yang terlihat pada rancangan desain pada Gambar 4.28 berikut :

The screenshot shows a web-based application interface. At the top, there's a navigation bar with links for 'Home', 'Data Klimatologi', 'Riwayat Proyeksi' (which is highlighted in blue), and 'Admin'. On the right side of the top bar, there are 'Logout' and 'Admin' dropdown menus. Below the navigation bar, there are two tabs: 'Proyeksi' (selected) and 'Buat Proyeksi'. The main content area is titled 'Proyeksi Hyperparameter' and contains several input fields arranged in two columns. The left column includes fields for 'Panjang Timestep' (value: 2), 'Max Epoch' (value: 50), 'Ukuran Batch' (value: 1), 'Jumlah Hidden Layers' (value: 1), and 'Jumlah Units' (value: 1). The right column includes fields for 'Nilai Learning Rate' (value: 0.1), 'Nilai Dropout' (value: 0.0), 'Tanggal Mulai Training' (value: 01/01/1985), 'Tanggal Akhir Training' (value: 12/31/2021), and 'Jumlah Prediksi ke Depan' (value: 5). At the bottom right of the form is a 'Kirim' button.

**Gambar 4.29. Desain *Form* Buat Proyeksi**

Pada Gambar 4.28 terdapat *input form* tanggal bertipe *datepicker* dan selainnya merupakan *input form text* dan terdapat 1 tombol kirim untuk mengirimkan nilai *hyperparameter* yang sudah di inputkan sehingga bisa di kirim permintaan ke *server* untuk di lakukan sebuah proyeksi.

#### 4.2.2.8. Desain Hasil Proyeksi Data

Setelah *admin* membuat opsi untuk prediksi dan klik tombol kirim seperti yang terlihat pada Gambar 4.28 *admin* di harapkan untung menunggu beberapa menit hingga proses *training* model dan prediksi mode selesai hingga mendapatkan respons dari *server*. Seperti yang terlihat pada rancangan desain pada Gambar 4.30 berikut :

The screenshot shows a web-based project management or data analysis interface. At the top, there's a navigation bar with links for Home, Data Klimatologi, Riwayat Proyeksi (selected), Admin, and Logout. Below the navigation is a sub-menu with Proyeksi and Buat Proyeksi options. The main content area is titled 'Hasil Proyeksi Data' and contains a table of hyperparameters:

Opsi	Value	Training Model....
Panjang Timestep	2	Predict Selected: : rr
Max Epoch	50	X Train Shape : (... , ... , ...)
Ukuran Batch	1	Y Train Shape : (... , ...)
Jumlah Hidden Layers	1	
Jumlah Units	1	Starting Training with Tensor 3D (N, F, S) : (... , ... , ...)
Nilai Learning Rate	0.1	
Nilai Dropout	0.0	Start Epoch 1/50
Jumlah Prediksi ke Depan	Jan 1, 1985	Training Batch .../...   [██████████] 100.0% Complete - loss: 0.09876
Jumlah Prediksi ke Depan	Dec 31, 2021	
Jumlah Prediksi ke Depan	5	

A vertical scrollbar is visible on the right side of the content area.

**Gambar 4.30. Desain Hasil Proyeksi Data**

Pada Gambar 4.30 respons hasil Proyeksi terlihat dalam bentuk detail nilai *hyperparameter* yang di inputkan sebelumnya dan juga catatan garis waktu proses algoritma Proyeksi berjalan mulai dari *preprocessing* data, *training* model, *testing* model, Prediksi model. Untuk hasil akhir akan di tampilkan bentuk grafik statistik sehingga lebih menarik untuk di lihat dan di pahami, mulai dari data asli, hasil *training*, hasil *testing*, maupun hasil Prediksi, nilai *error* hasil *training* dan *testing* data juga di tampilkan pada bagian akhir dari halaman hasil Proyeksi.

## **BAB V**

### **IMPLEMENTASI DAN PENGUJIAN**

#### **5.1. Implementasi Sistem**

Implementasi Sistem merupakan tahap dalam alur siklus hidup mengembangkan sistem. Sebuah implementasi diperlukan agar perancangan *interface* dan penelitian kode program sesuai dengan sistem yang dirancang ataupun yang telah di analisa sebelumnya.

Untuk melakukan atau mengimplementasikan program aplikasi yang telah dirancang, maka diperlukan sebuah alat bantu berupa komputer, yang mana untuk mengoperasikan komputer itu sendiri yang memerlukan tiga buah komponen pendukung seperti *hardware*, *software*, dan *brainware*.

##### **5.1.1. *Hardware***

*Hardware* yang di gunakan dalam implementasikan program yang telah di rancang berupa satu *unit* komputer atau laptop yang lengkap keseluruhan perangkatnya. Berikut Spesifikasi minimum dari Perangkat Keras / *Hardware* yang di butuhkah dalam menjalankan sistem sebagai berikut :

- a) CPU Quad Core @ 2.00GHz, atau lebih
- b) *Memory* RAM 3GB, atau lebih
- c) Partisi Penyimpanan 20GB, atau lebih

##### **5.1.2. *Software***

Untuk menjalankan sistem yang dirancang harus menggunakan beberapa dependensi dan *software* pendukung, beberapa *software* pendukung yang harus di

instal berfungsi untuk tempat menjalankan sistem tersebut. Berikut beberapa Versi Perangkat Lunak / *Software* yang di butuhkah dalam menjalankan sistem sebagai berikut :

- a) Sistem Operasi Windows 7 64-bit, atau lebih
- b) Bahasa Pemrograman Python v.3.6.0 s/d v.3.9.0
- c) MySQL Ver 8.0, atau lebih

#### **5.1.3. *Brainware***

*Brainware* merupakan operator yang berfungsi untuk mengoperasikan atau menjalankan program. Jadi ketiga komponen di atas memiliki komponen abstrak dari susunan sistem komputer dan *hardware* dan memiliki fungsi jika digunakan bersama-sama dengan *software* sedangkan *brainware* adalah orang yang mengoperasikan program, tanpa *brainware* komputer tidak bisa beroperasi.

#### **5.1.4. Lingkungan Implementasi**

Dalam implementasi dan pengujian peneliti menggunakan beberapa perangkat keras dan perangkat lunak sebagai dependensi yang di gunakan sebagai berikut :

##### **5.1.4.1. Perangkat Keras (*Hardware*)**

Perangkat Keras yang di gunakan untuk proses implementasi dan pengujian sistem menggunakan perangkat laptop dengan spesifikasi sebagai berikut :

- a) Laptop ASUS Model A445LAB
- b) CPU Intel® Core™ i3-5005U CPU @ 2.00GHz (4 CPUs)
- c) *Memory* RAM 12GB
- d) Partisi Penyimpanan Samsung SSD 870 EVO 250GB

- e) GPU Intel(R) HD Graphics 5500

#### **5.1.4.2. Perangkat Lunak (*Software*)**

Perangkat Keras yang di gunakan untuk proses implementasi dan pengujian sistem menggunakan perangkat laptop dengan spesifikasi sebagai berikut :

- a) Sistem Operasi Windows 10 Pro 64bit (10.0, Build 19044)
- b) Google Chrome (64-bit)
- c) Bahasa Pemrograman Python v.3.9.0 (64-bit)
- d) MySQL Ver 8.0.27 (64-bit)

### **5.2. Proses Instalasi**

Dalam tahap implementasi dan perancangan sistem yang di rancang diperlukannya sebuah sistem operasi windows, *software* dan dependensi pendukung yang digunakan sebagai web *server* untuk mengetahui hasil dari sistem yang sudah dibuat.

#### **5.2.1. Tahap Instalasi Bahasa Pemograman Python**

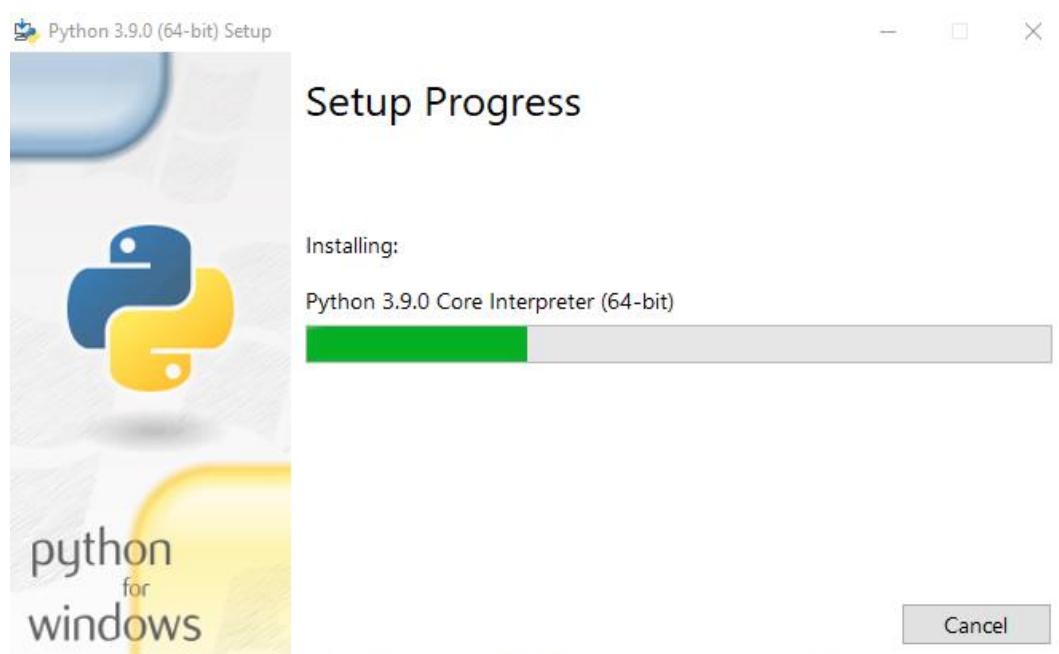
Python merupakan Bahasa pemrograman interpretatif, versi yang di gunakan merupakan versi 3.9.0 yang dapat di download website resmi python di [www.python.org](http://www.python.org), Perlu di lakukan instalasi dan konfigurasi terlebih dahulu agar bahasa pemrograman python dapat berjalan pada perangkat yang di gunakan. Adapun tahap instalasi dan konfigurasinya sebagai berikut :

- 1) Klik 2 kali pada file Windows Installer Package Python versi 3.9.0 yang sudah di download pada website resmi python, kemudian centang ***Install launcher for all users (recommended)***, dan centang juga ***Add Python 3.9 to PATH***, selanjutnya klik ***Install Now*** seperti Gambar 5.1 di bawah ini.



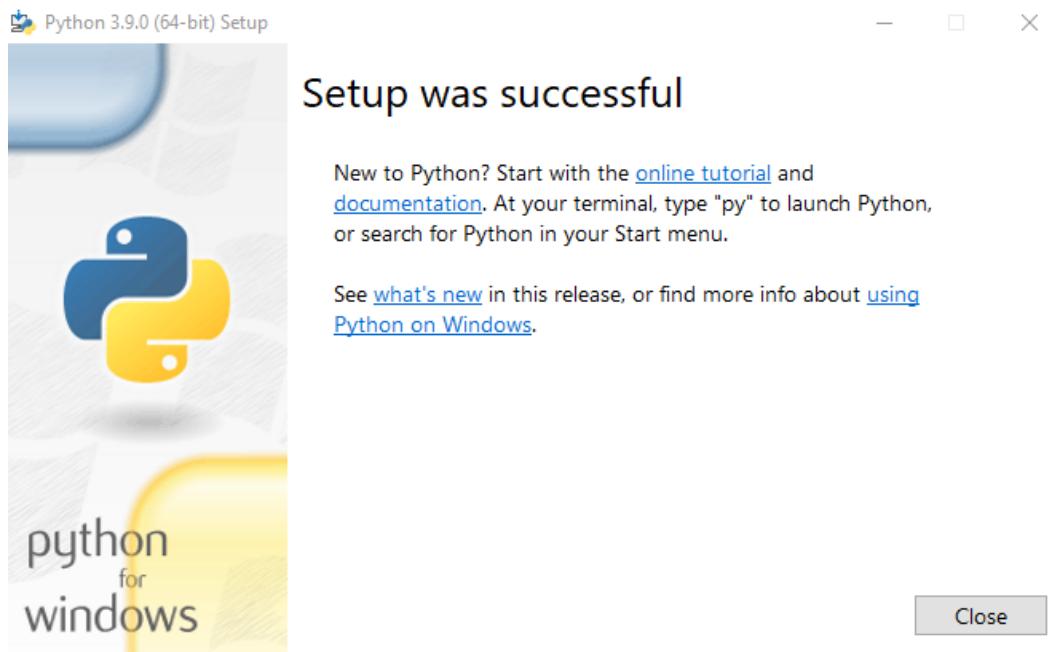
**Gambar 5.1. Antarmuka Awal Instalasi Python**

- 2) Kemudian tunggu hingga proses bar proses instalasi selesai, seperti Gambar 5.2 di bawah ini :



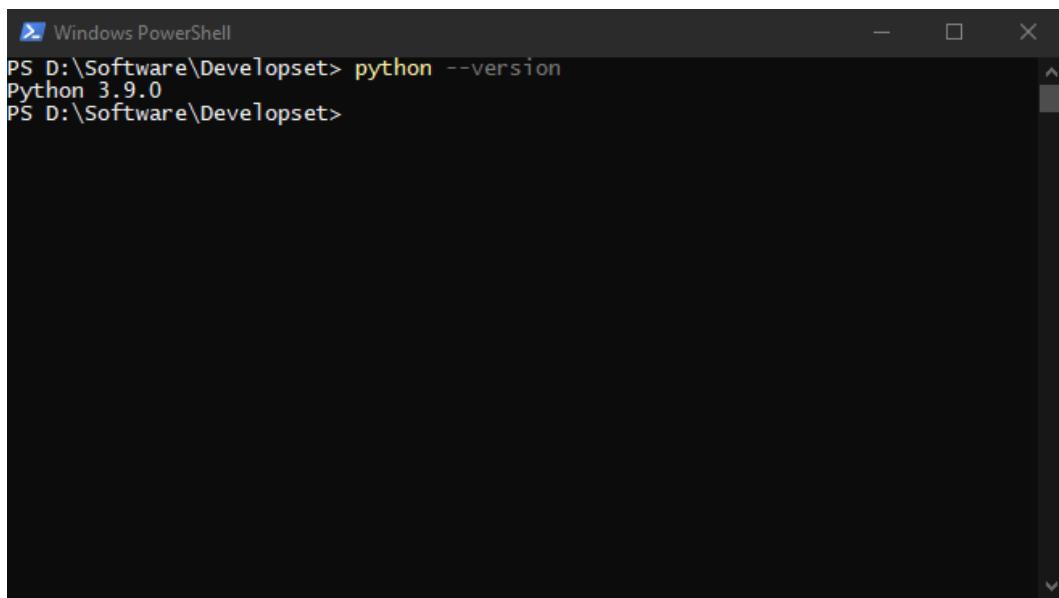
**Gambar 5.2. Antarmuka Bar Proses Instalasi Python**

- 3) Terakhir Ketika sukses tampil antarmuka instalasi seperti pada Gambar 5.3 di bawah ini :



**Gambar 5.3. Antarmuka Instalasi Python Berhasil**

- 4) Untuk memastikan Python sudah terinstal dan *Python PATH* sudah terdaftar pada *windows environment variables* dapat di lihat dengan memberikan sebuah perintah pada terminal windows seperti yang terlihat pada Gambar 5.4 di bawah ini :



```
PS D:\Software\Developset> python --version
Python 3.9.0
PS D:\Software\Developset>
```

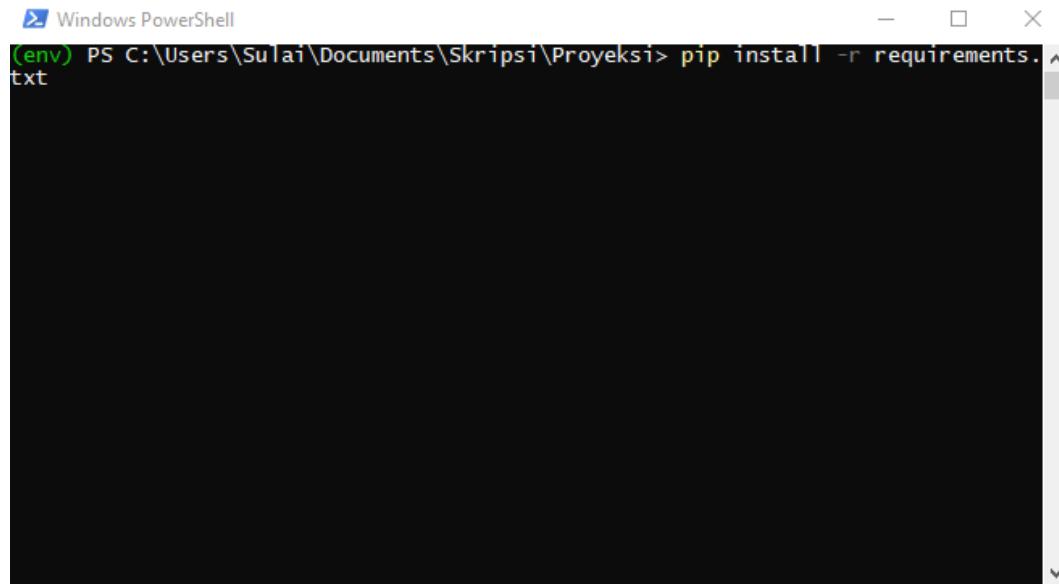
**Gambar 5.4. Perintah Melihat Versi Python yang Terinstall**

### 5.2.2. Tahap Instalasi Module Dependensi Python

Module pada python merupakan blok-blok kode yang memiliki peran peran tertentu biasanya memiliki nama tersendiri, yang bertugas untuk menyelesaikan satu set perintah tertentu, bisa di anggap sebagai pekakas tertentu yang dapat di panggil dari bagian program manapun. Adapun tahap instalasi dan konfigurasinya sebagai berikut :

- 1) Pertama persiapkan terminal yang di gunakan, sebagai contoh peneliti di sini menggunakan terminal powershell bawaan windows.
- 2) Kemudian masuk ke direktori tempat program sistem di simpan, contohnya peneliti meletakan program sistem pada lokasi direktori **“C:\Users\Sulai\Documents\Skripsi\Proyeksi”**
- 3) Lalu ketika terminal sudah berada di direktori tempat program sistem berada, ketik perintah **“pip install -r requirements.txt”** dan pastikan terdapat file

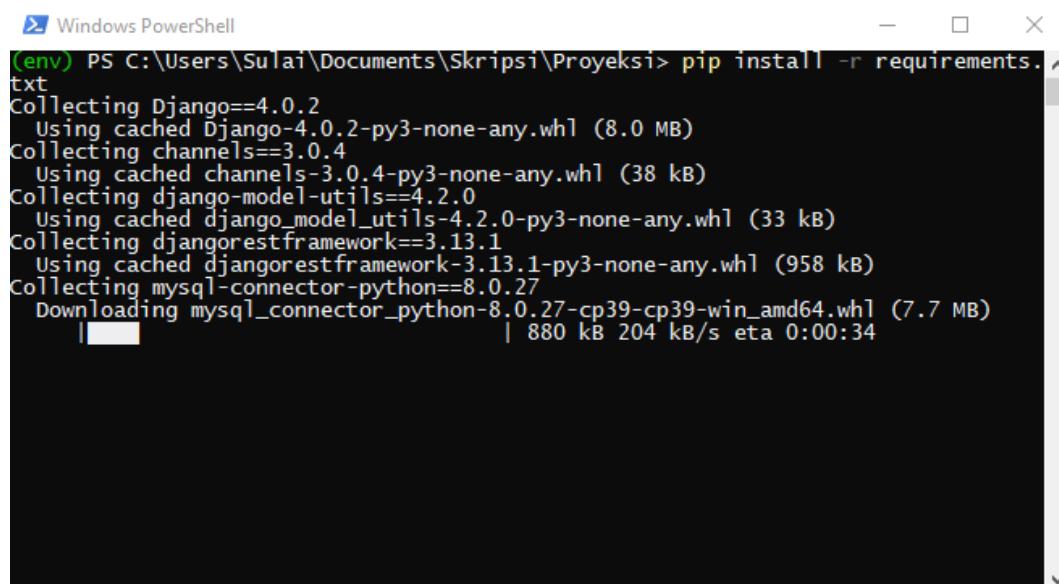
Bernama “*requirements.txt*” di direktori dimana termpat terminal mengeksekusi perintah, seperti yang terlihat pada Gambar 5.5 berikut :



```
(env) PS C:\Users\Sulai\Documents\Skripsi\Proyeksi> pip install -r requirements.txt
```

**Gambar 5.5. Perintah Untuk Menginstall Module Python yang di Butuhkan**

- 5) Kemudian muncul tampilan installasi module, tunggu hingga proses instalasi selesai, seperti Gambar 5.6 di bawah ini :



```
(env) PS C:\Users\Sulai\Documents\Skripsi\Proyeksi> pip install -r requirements.txt
Collecting Django==4.0.2
  Using cached Django-4.0.2-py3-none-any.whl (8.0 MB)
Collecting channels==3.0.4
  Using cached channels-3.0.4-py3-none-any.whl (38 kB)
Collecting django-model-utils==4.2.0
  Using cached django_model_utils-4.2.0-py3-none-any.whl (33 kB)
Collecting djangorestframework==3.13.1
  Using cached djangorestframework-3.13.1-py3-none-any.whl (958 kB)
Collecting mysql-connector-python==8.0.27
  Downloading mysql_connector_python-8.0.27-cp39-cp39-win_amd64.whl (7.7 MB)
    |██████████| 880 kB 204 kB/s eta 0:00:34
```

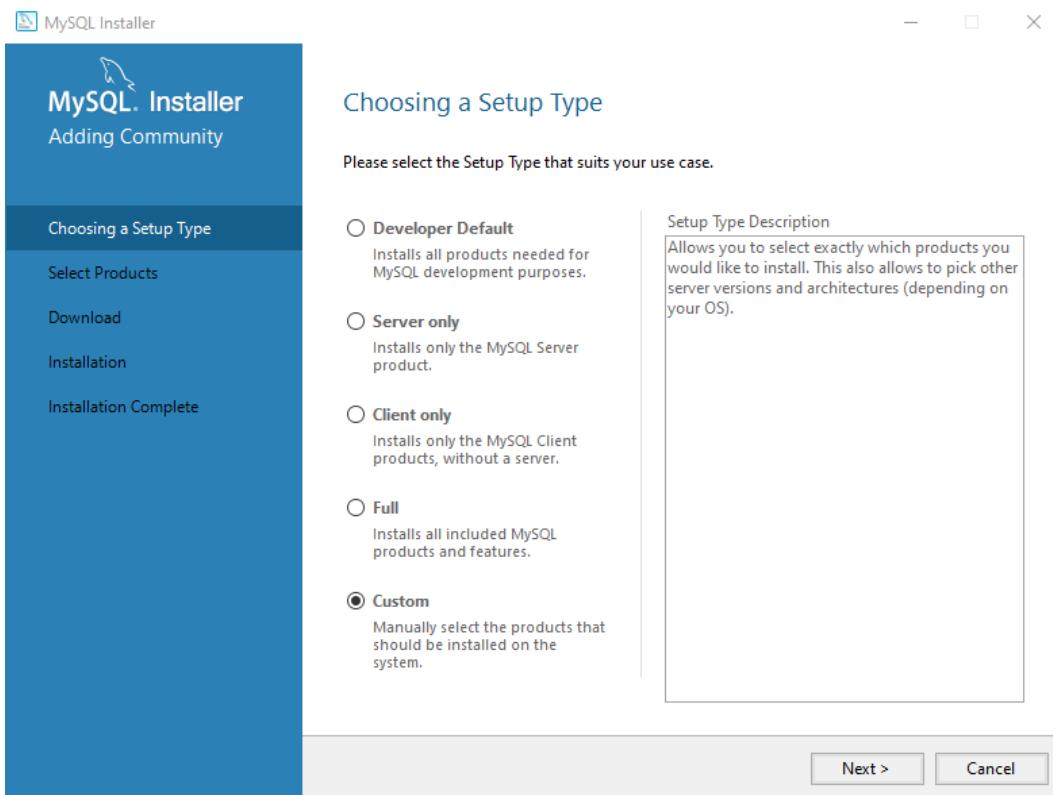
**Gambar 5.6. Python Melakukan Installasi Module Yang di Butuhkan**

### 5.2.3. Tahap Instalasi MySQL

Python merupakan Bahasa pemrograman interpretatif, versi yang di gunakan merupakan versi 3.9.0 yang dapat di download website resmi python di dev.mysql.com, Perlu di lakukan instalasi dan konfigurasi terlebih dahulu agar bahasa pemrograman python dapat berjalan pada perangkat yang di gunakan.

Adapun tahap instalasi dan konfigurasinya sebagai berikut :

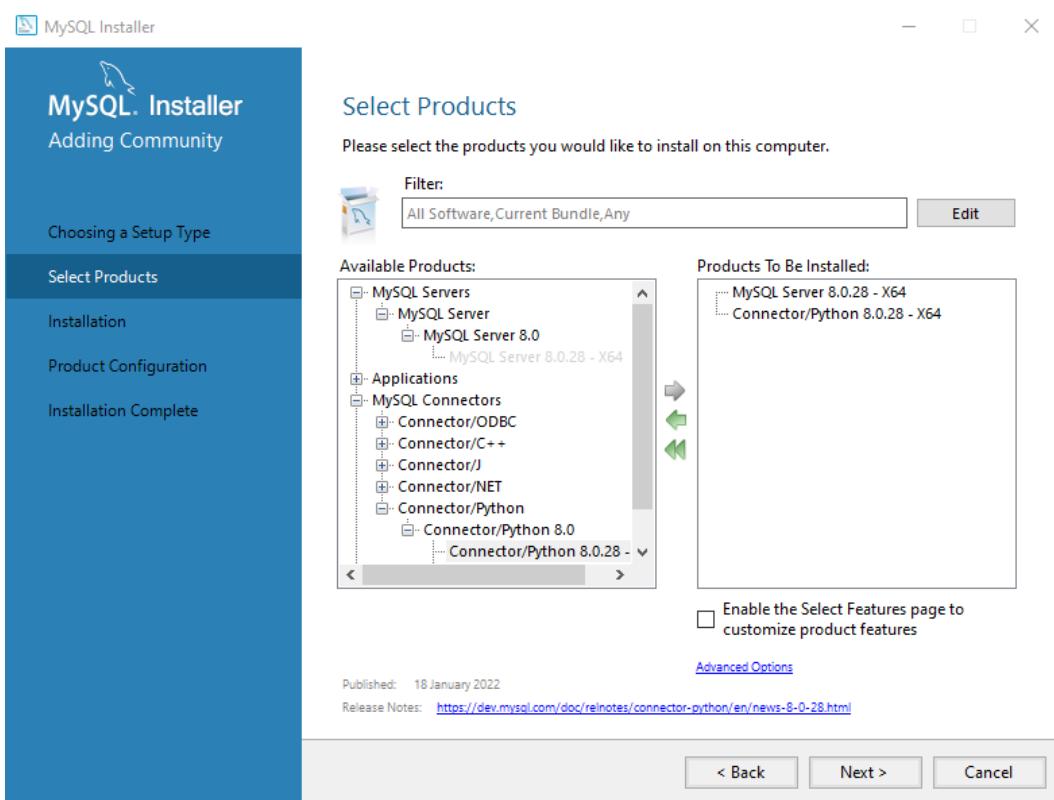
- 1) Klik 2 kali pada file Windows Installer Package MySQL versi 8.023.0 yang sudah di download pada website resmi MySQL, kemudian centang ***Install launcher for all users (recommended)***, dan centang juga ***Add Python 3.9 to PATH***, selanjutnya klik ***Install Now*** seperti Gambar 5.1 di bawah ini.



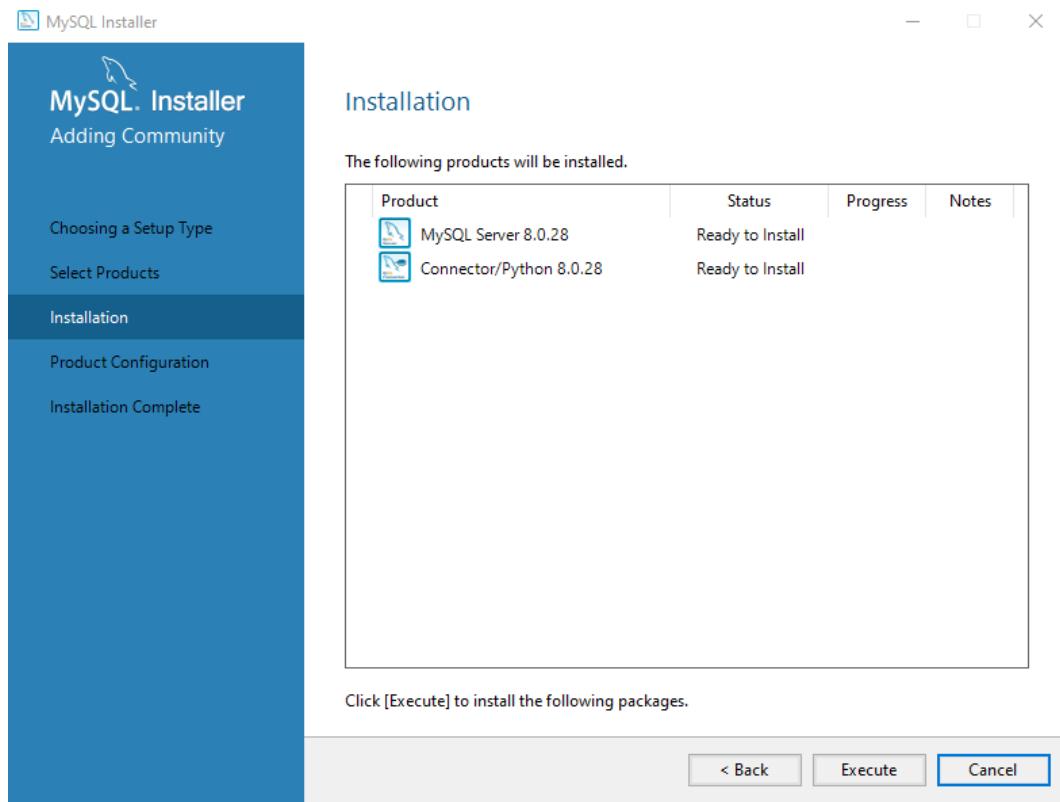
**Gambar 5.7. awdawda**

- 2) Klik 2 kali pada file

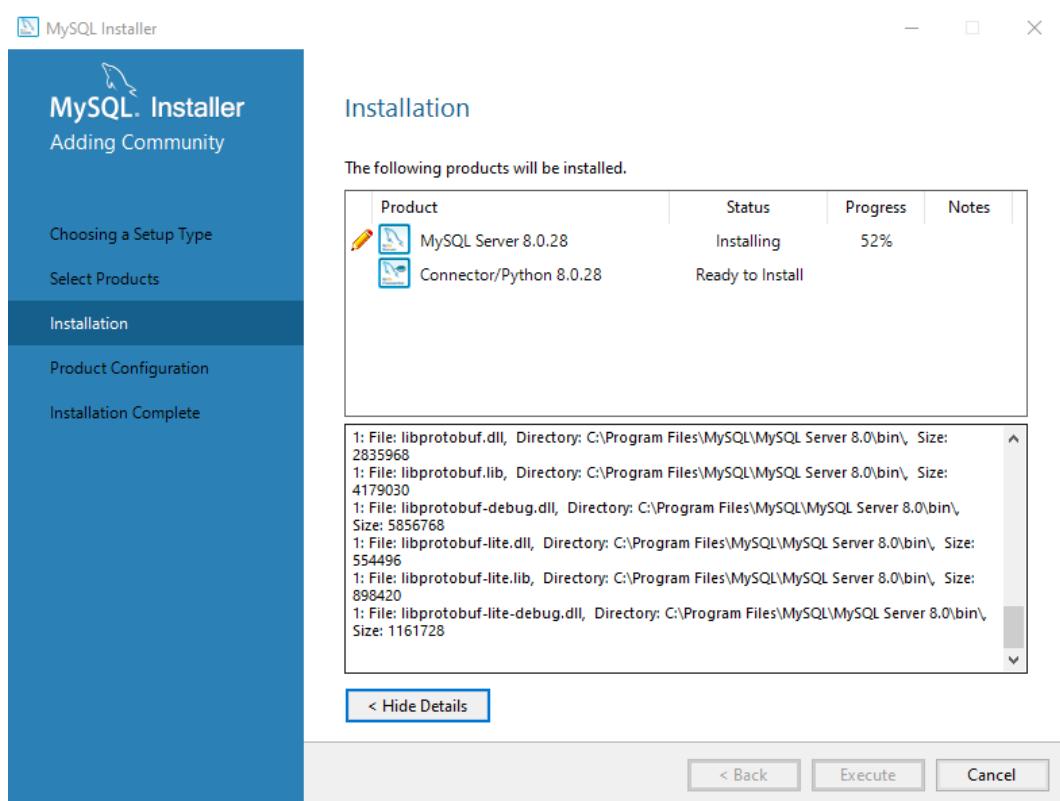
Awdawdaw



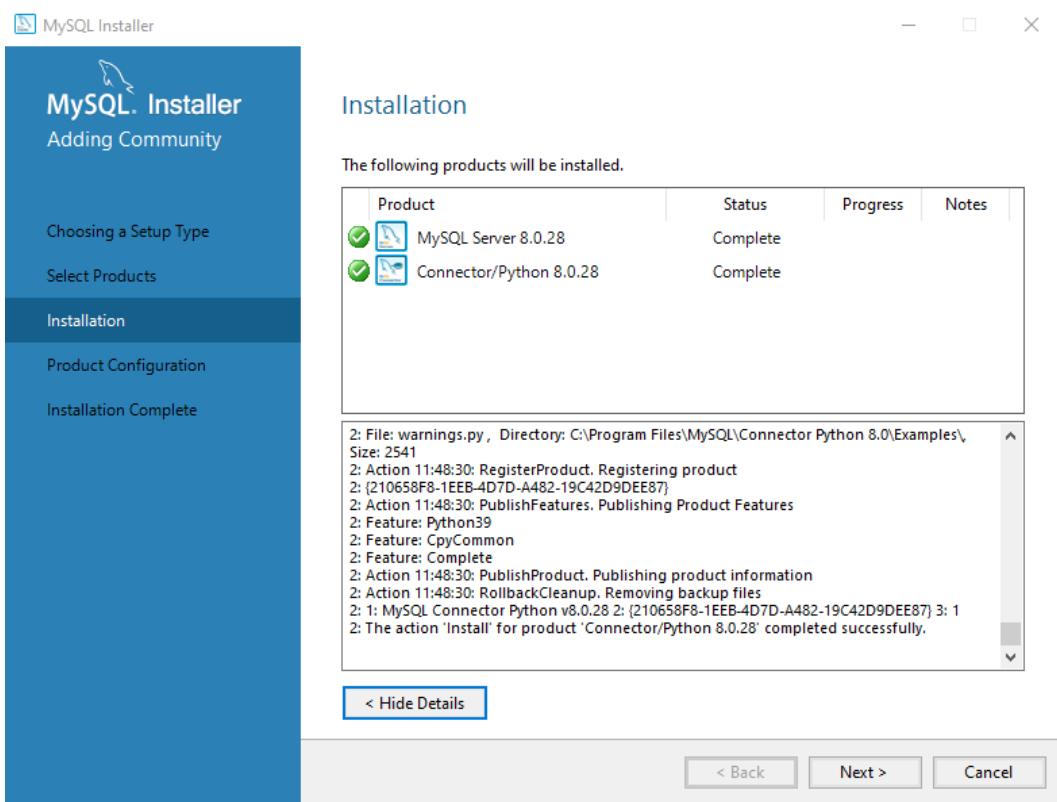
3) Awdawdawdawda



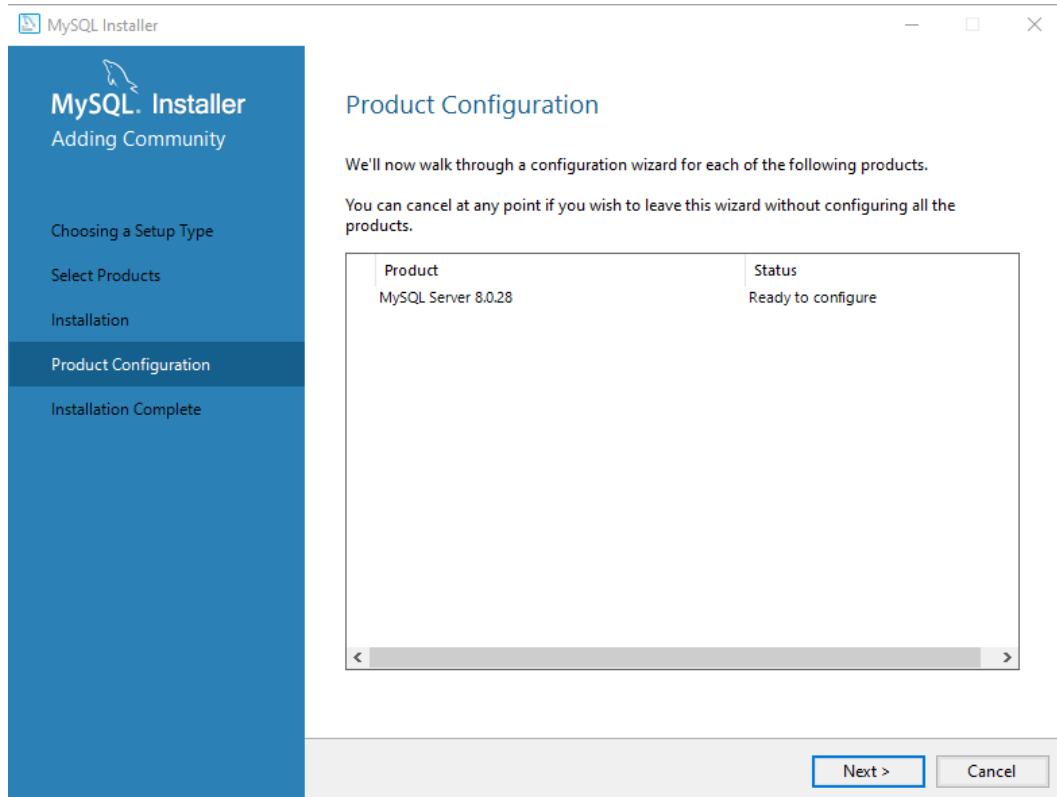
#### 4) Awdawd



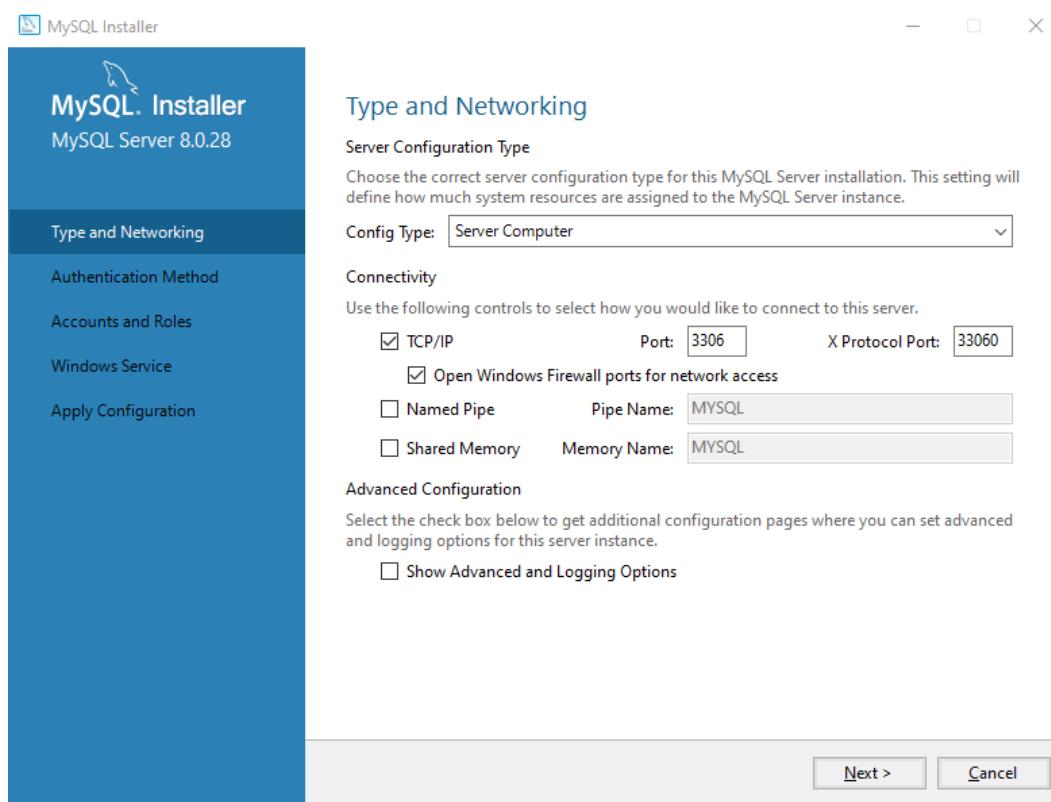
## 5) Awdaw



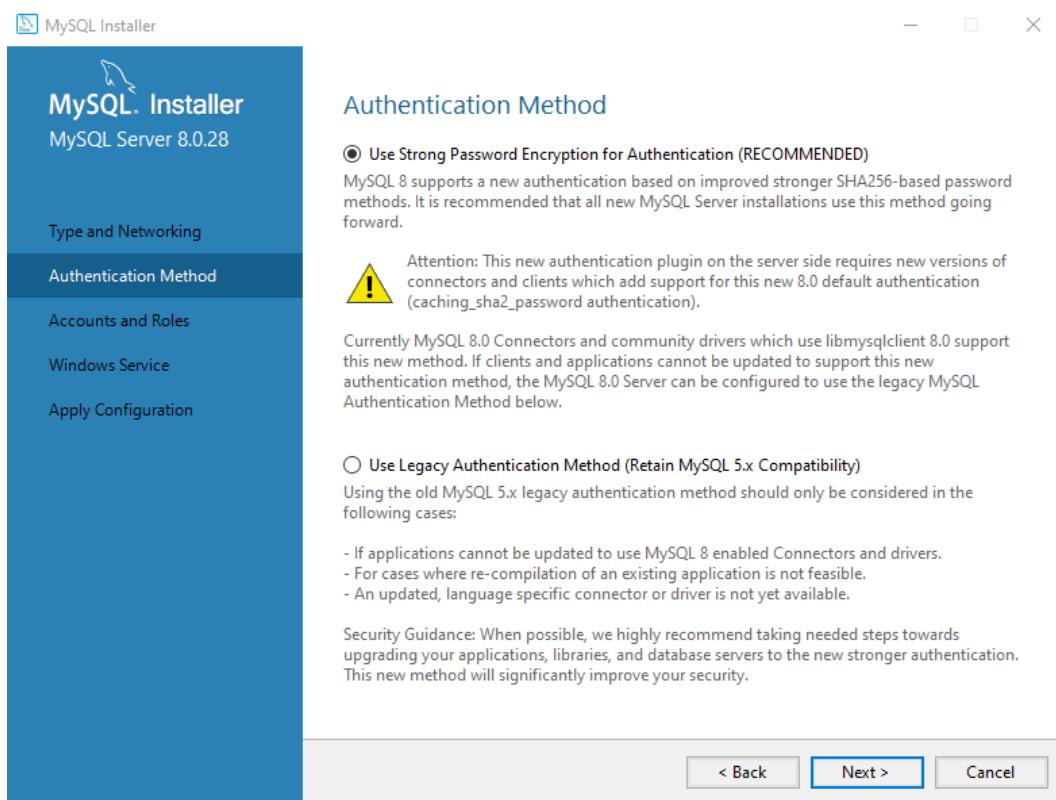
## 6) Awdawda



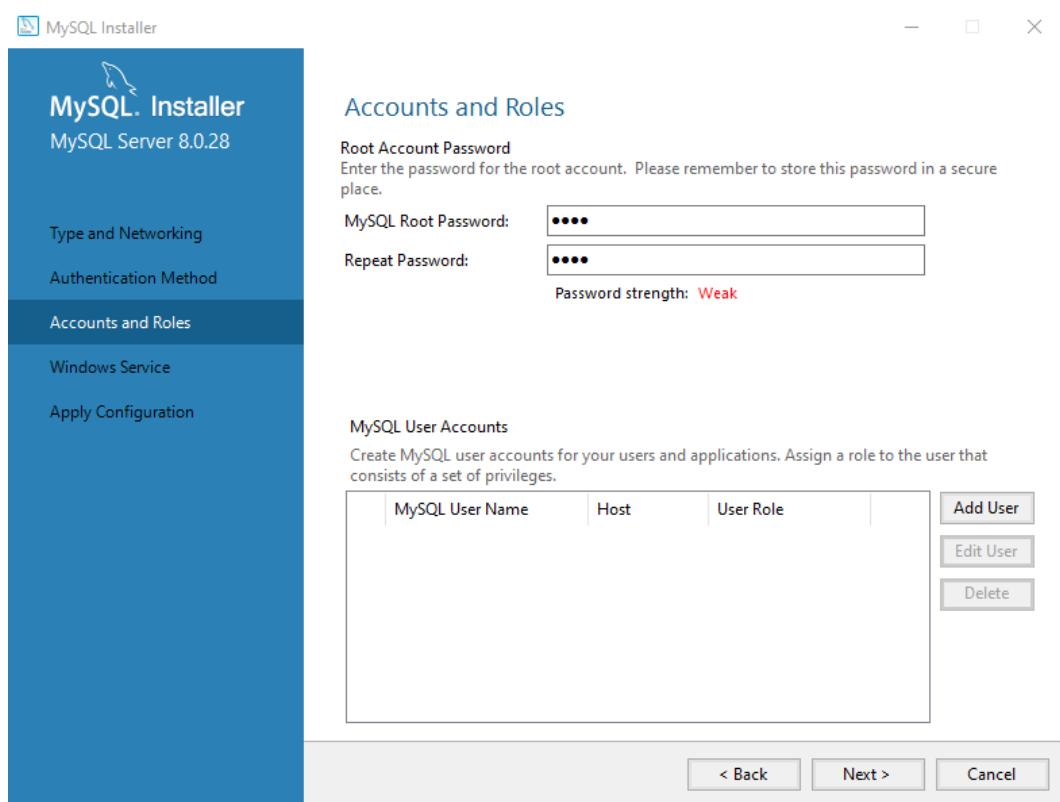
## 7) Awdawdawd



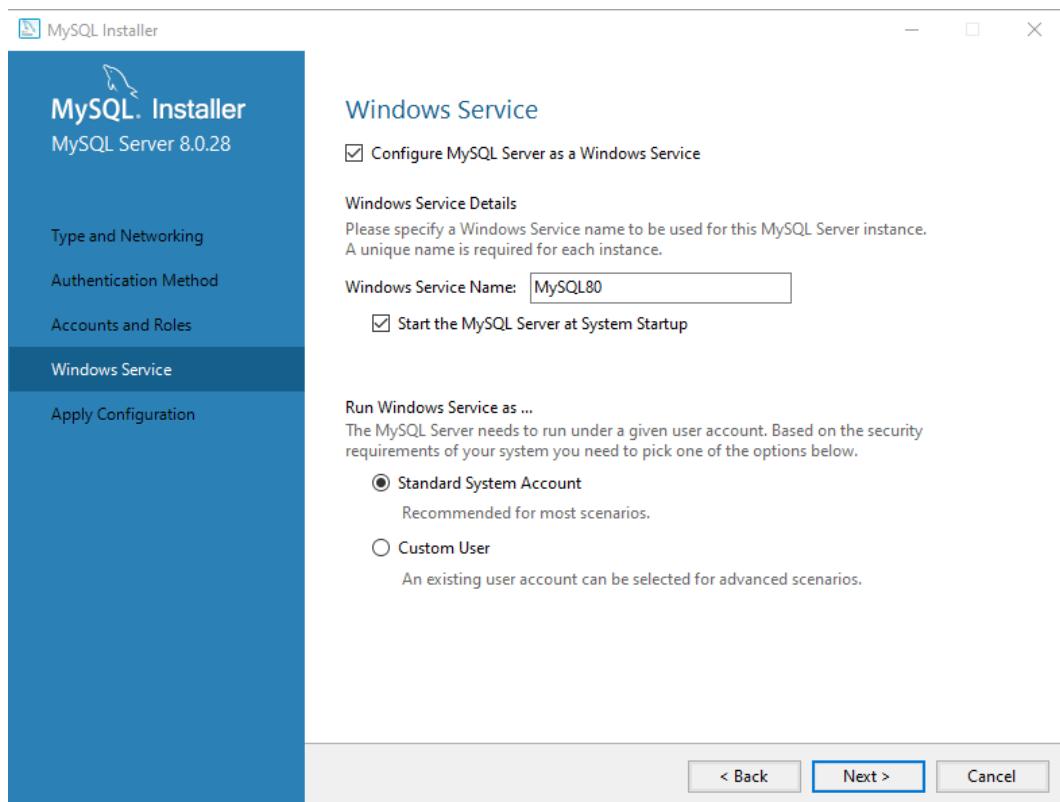
## 8) Awdawdawda



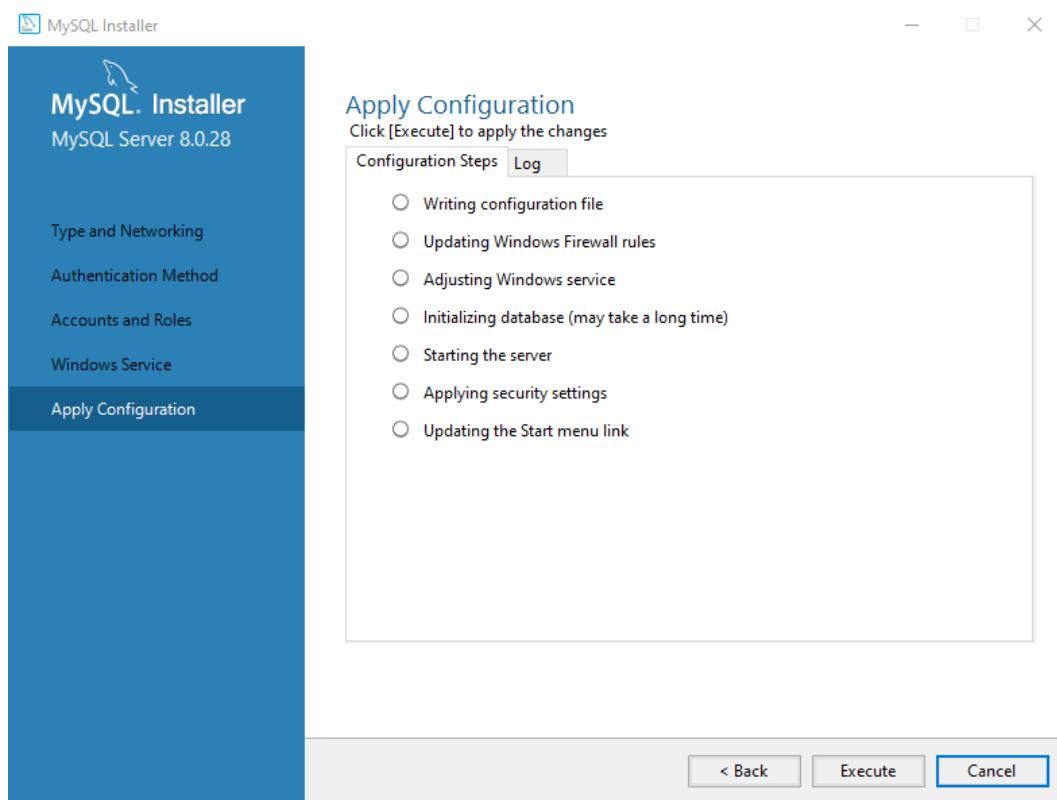
## 9) Awdawdawda



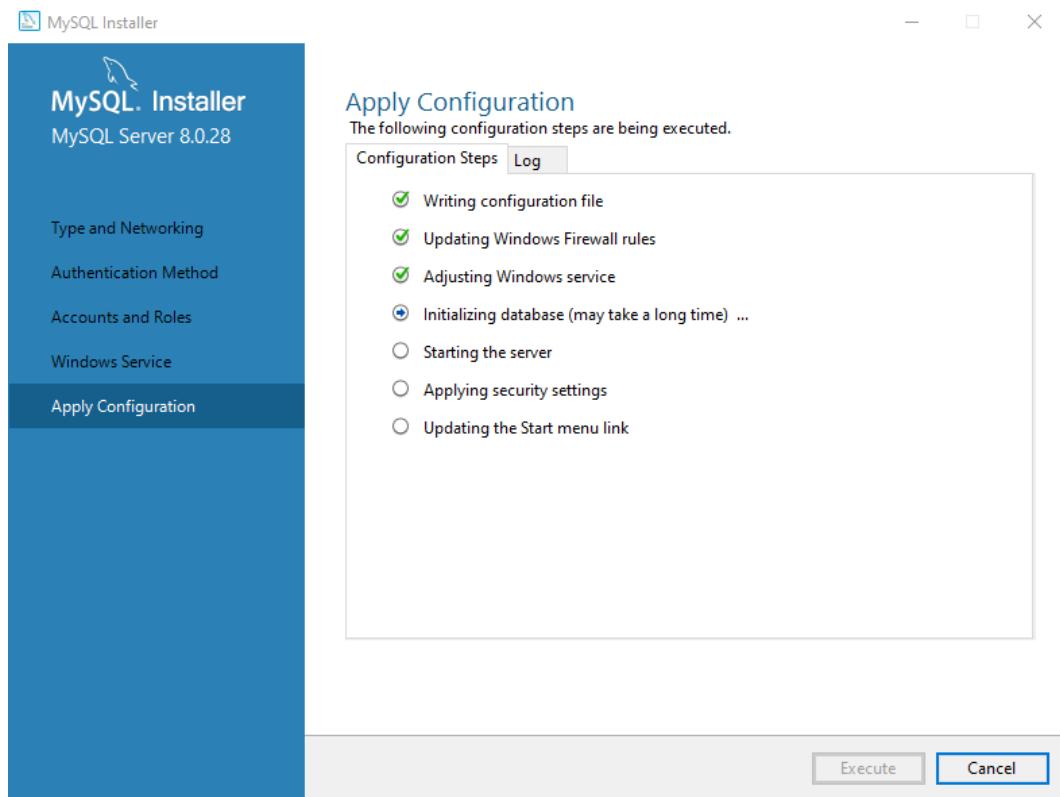
## 10) Awdawdawda



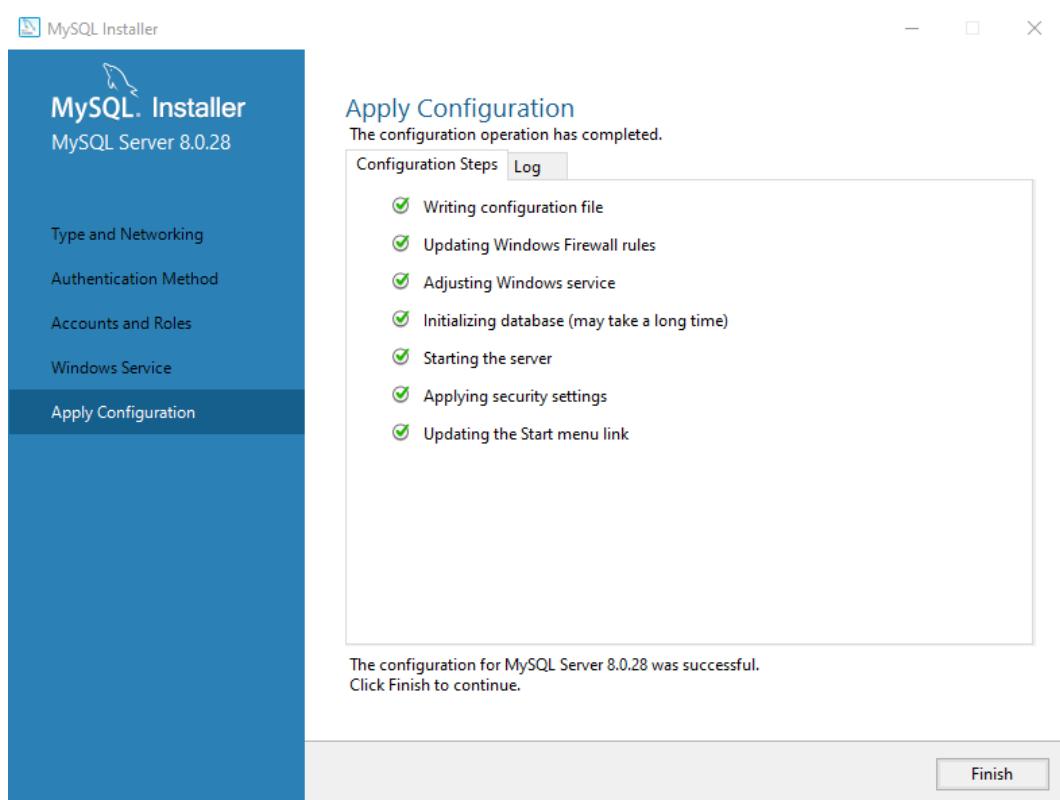
11) Awdawda



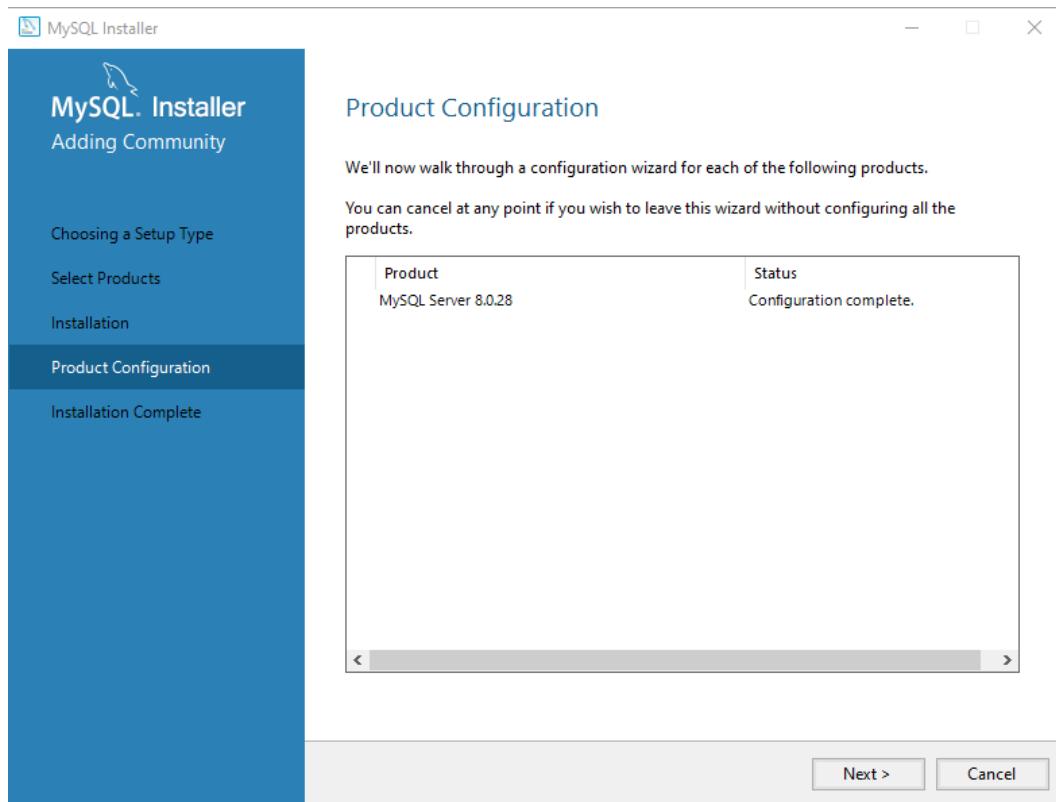
12) Awdawda



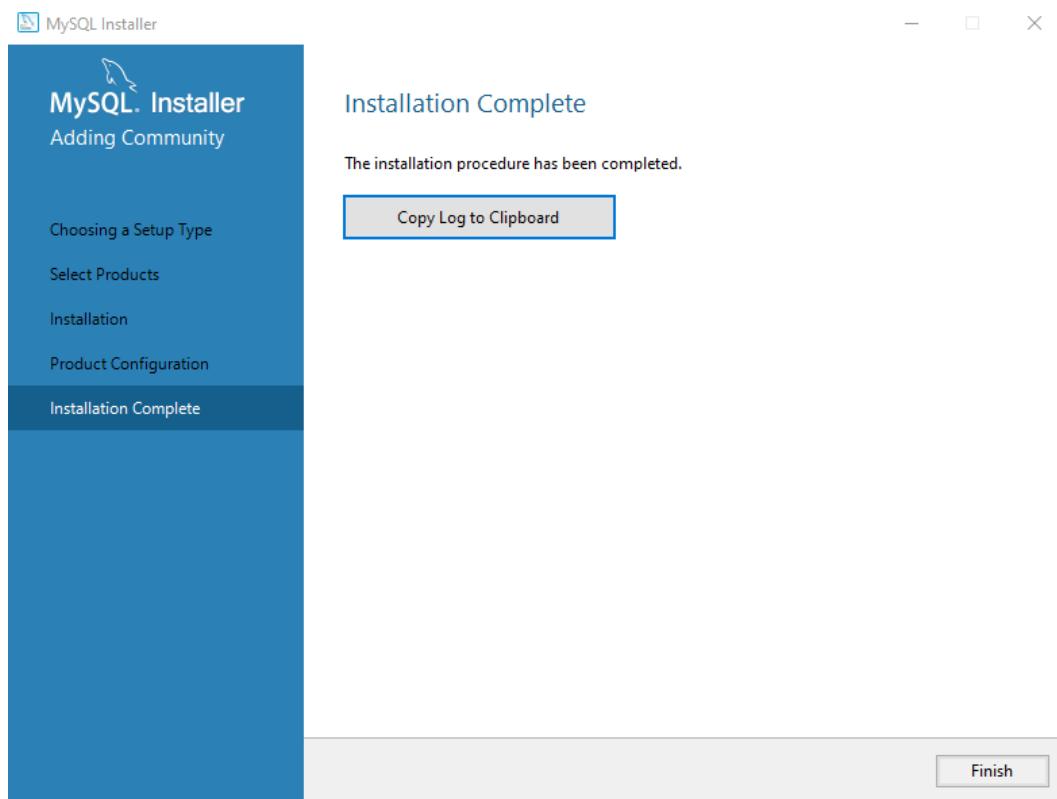
### 13) Awdawda



14) Awdawd



15) Awdawda

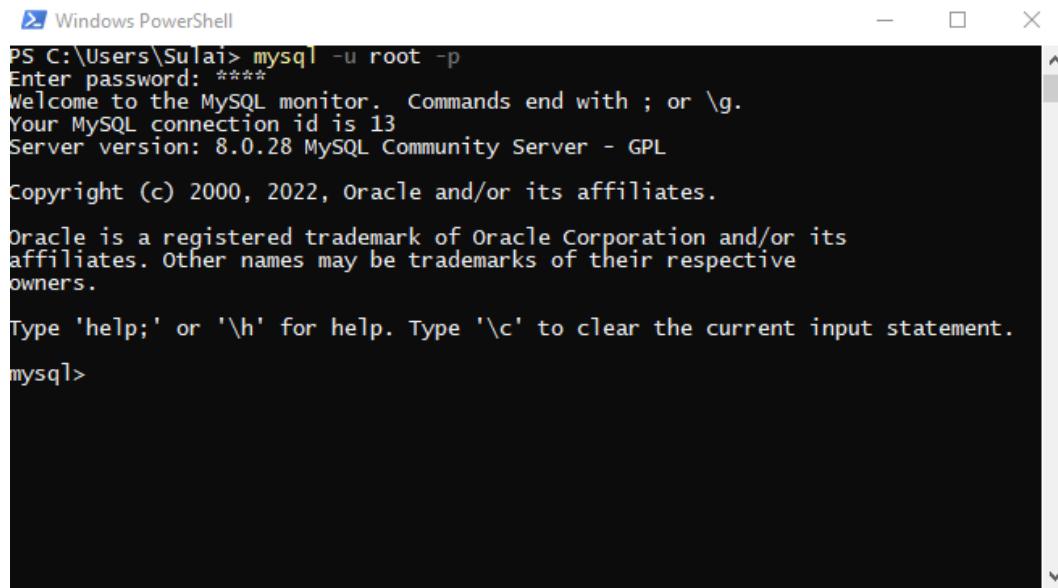


16) Awdawda

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command "mysql --version" was run, and the output shows "Ver 8.0.28 for Win64 on x86\_64 (MySQL Community Server - GPL)".

```
PS C:\Users\Sulai> mysql --version
C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe Ver 8.0.28 for Win64 on x
86_64 (MySQL Community Server - GPL)
PS C:\Users\Sulai>
```

17) Awdawda



```
PS C:\Users\Sulai> mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.28 MySQL Community Server - GPL

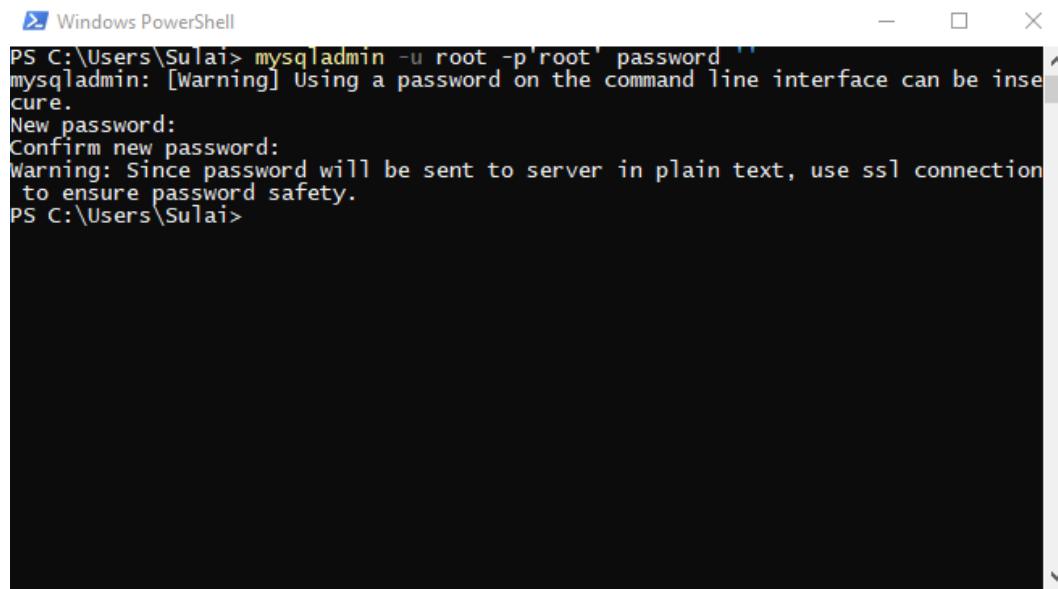
Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

18) Awdawd



```
PS C:\Users\Sulai> mysqladmin -u root -p'root' password ''
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
New password:
Confirm new password:
Warning: Since password will be sent to server in plain text, use ssl connection
to ensure password safety.
PS C:\Users\Sulai>
```

19) Awdawda

20) awdawda

### 5.3. Pengujian *Interface*

Dalam melakukan sebuah implementasi maka diperlukan program komputer yaitu perancangan *interface* dan penelitian kode program sesuai dengan sistem yang dirancang.

Sistem dapat dilakukan setelah merancang sistem beserta *interface*-nya. Perancangan *interface* dilakukan untuk interaksi *user* dengan sistem yang telah dibuat. Dalam melakukan implementasi sistem dibutuhkan program dan penelitian program (*scripting*) yang sesuai dengan sistem yang dirancang.

Implementasi sistem dapat dilakukan setelah sistem yang dibuat dapat berjalan sebagaimana mestinya. Untuk itu pada bab ini dijelaskan bentuk asli dari tampilan sebenarnya apabila sistem ini diakses oleh *user*.

Melakukan sebuah implementasi maka diperlukan program komputer yaitu perancangan *interface* dan penelitian kode program sesuai dengan sistem yang dirancang.